Journal of Global Optimization **11:** 341–359, 1997.

Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces

RAINER STORN
Siemens AG, ZFE T SN2, Otto-Hahn Ring 6, D-81739 Muenchen, Germany.
(e-mail:rainer.storn@mchp.siemens.de)

KENNETH PRICE
836 Owl Circle, Vacaville, CA 95687, U.S.A. (email: kprice@solano.community.net)

(Received: 20 March 1996; accepted: 19 November 1996)

Abstract. A new heuristic approach for minimizing possibly nonlinear and non-differentiable continuous space functions is presented. By means of an extensive testbed it is demonstrated that the new method converges faster and with more certainty than many other acclaimed global optimization methods. The new method requires few control variables, is robust, easy to use, and lends itself very well to parallel computation.

Key words: Stochastic optimization, nonlinear optimization, global optimization, genetic algorithm, evolution strategy.

1. Introduction

Problems which involve global optimization over continuous spaces are ubiquitous throughout the scientific community. In general, the task is to optimize certain properties of a system by pertinently choosing the system parameters. For convenience, a system's parameters are usually represented as a vector. The standard approach to an optimization problem begins by designing an objective function that can model the problem's objectives while incorporating any constraints. Especially in the circuit design community, methods are in use which do not need an objective function but operate with so-called regions of acceptability: Brayton *et al*. (1981), Lueder (1990), Storn (1995). Although these methods can make formulating a problem simpler, they are usually inferior to techniques which make use of an objective function. Consequently, we will only concern ourselves with optimization methods that use an objective function. In most cases, the objective function defines the optimization problem as a minimization task. To this end, the following investigation is further restricted to minimization problems. For such problems, the objective function is more accurately called a "cost" function.

When the cost function is nonlinear and non-differentiable, direct search approaches are the methods of choice. The best known of these are the algorithms

by Nelder and Mead: Bunday *et al*. (1987), by Hooke and Jeeves: Bunday *et al*. (1987), genetic algorithms (GAs): Goldberg (1989), and evolution strategies (ESs): Rechenberg (1973), Schwefel (1995). Central to every direct search method is a strategy that generates variations of the parameter vectors. Once a variation is generated, a decision must then be made whether or not to accept the newly derived parameters. Most standard direct search methods use the greedy criterion to make this decision. Under the greedy criterion, a new parameter vector is accepted if and only if it reduces the value of the cost function. Although the greedy decision process converges fairly fast, it runs the risk of becoming trapped in a local minimum. Inherently parallel search techniques like genetic algorithms and evolution strategies have some built-in safeguards to forestall misconvergence. By running several vectors simultaneously, superior parameter configurations can help other vectors escape local minima. Another method which can extricate a parameter vector from a local minimum is Simulated Annealing: Ingber (1992), Ingber (1993), Press *et al*. (1992). Annealing relaxes the greedy criterion by occasionally permitting an uphill move. Such moves potentially allow a parameter vector to climb out of a local minimum. As the number of iterations increases, the probability of accepting a large uphill move decreases. In the long run, this leads to the greedy criterion. While all direct search methods lend themselves to annealing, it has primarily been used just for the Random Walk, which itself is the simplest case of an evolutionary algorithm: Rechenberg (1973). Nevertheless, attempts have been made to anneal other direct searches like the method of Nelder and Mead: Press *et al*. (1992) and genetic algorithms: Ingber (1993), Price (1994). Users generally demand that a practical minimization technique should fulfill five requirements:

(1) Ability to handle non-differentiable, nonlinear and multimodal cost functions.

(2) Parallelizability to cope with computation intensive cost functions.

(3) Ease of use, i.e. few control variables to steer the minimization. These variables should also be robust and easy to choose.

(4) Good convergence properties, i.e. consistent convergence to the global minimum in consecutive independent trials.

As explained in the following the novel minimization method Differential Evolution (DE) was designed to fulfill all of the above requirements.

To fulfill requirement (1) DE was designed to be a stochastic direct search method. Direct search methods also have the advantage of being easily applied to experimental minimization where the cost value is derived from a physical experiment rather than a computer simulation. Physical experiments were indeed the motivation for the development of ESs by Rechenberg et al.

Requirement (2) is important for computationally demanding optimizations where, for example, one evaluation of the cost function might take from minutes to hours, as is often the case in integrated circuit design or finite element simulation. In order to obtain usable results in a reasonable amount of time, the only viable approach is to resort to a parallel computer or a network of computers. DE fulfills

requirement (2) by using a vector population where the stochastic perturbation of the population vectors can be done independently.

In order to satisfy requirement (3) it is advantageous if the minimization method is self-organizing so that very little input is required from the user. The method of Nelder and Mead: Bunday *et al*. (1987), is a good example of a self-organizing minimizer. If the cost function at hand has D parameters Nelder and Mead's method uses a polyhedron with D+1 vertices to define the current search space. Each vertex is represented by a D-dimensional parameter vector which samples the cost function. New parameter vectors are generated by so-called reflections of the vectors with highest cost and contractions around low cost vectors. The new vectors replace their predecessors if they correspond to a function value with reduced cost compared to their predecessors. This strategy allows the search space, i.e. the polyhedron, to expand and contract without special control variable settings by the user. Unfortunately, Nelder & Mead's method is basically a local minimization method, which, as our experiments suggest, is not powerful enough for global minimization tasks, even if the concept of annealing is introduced. Yet DE borrows the idea from Nelder & Mead of employing information from within the vector population to alter the search space. DE's self-organizing scheme takes the difference vector of two randomly chosen population vectors to perturb an existing vector. The perturbation is done for every population vector. This crucial idea is in contrast to the method used by traditional ESs in which predetermined probability distribution functions determine vector perturbations.

Last but not least, the good convergence properties demanded in requirement (4) are mandatory for a good minimization algorithm. Although many approaches exist to theoretically describe the convergence properties of a global minimization method, only extensive testing under various conditions can show whether a minimization method can fulfill its promises. DE scores very well in this regard as will be explained in detail in Section 3.

2. Differential Evolution

Differential Evolution (DE) is a parallel direct search method which utilizes NP D-dimensional parameter vectors

$$x_{i,G}, i = 1, 2, \ldots, \text{NP} \tag{1}$$

as a population for each generation G. NP does not change during the minimization process. The initial vector population is chosen randomly and should cover the entire parameter space. As a rule, we will assume a uniform probability distribution for all random decisions unless otherwise stated. In case a preliminary solution is available, the initial population might be generated by adding normally distributed random deviations to the nominal solution $x_{\text{nom},0}$. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. Let this operation be called mutation. The mutated vector's parameters

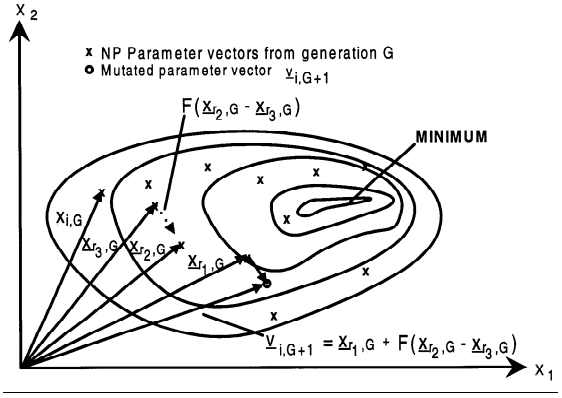

Figure 1. An example of a two-dimensional cost function showing its contour lines and the process for generating $v_{i,G+1}$.

are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. Parameter mixing is often referred to as "crossover" in the ES-community and will be explained later in more detail. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. Each population vector has to serve once as the target vector so that NP competitions take place in one generation.

More specifically DE's basic strategy can be described as follows:

Mutation

For each target vector $x_{i,G}, i = 1, 2, 3, \ldots,$ NP, a mutant vector is generated according to

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{2}$$

with random indexes $r_1, r_2, r_3 \in \{1, 2, \ldots, \text{NP}\}$, integer, mutually different and $F > 0$. The randomly chosen integers r_1, r_2 and r_3 are also chosen to be different from the running index i, so that NP must be greater or equal to four to allow for this condition. F is a real and constant factor $\in [0, 2]$ which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$. Figure 1 shows a two-dimensional example that illustrates the different vectors which play a part in the generation of $v_{i,G+1}$.

Crossover

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. To this end, the trial vector:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}) \tag{3}$$

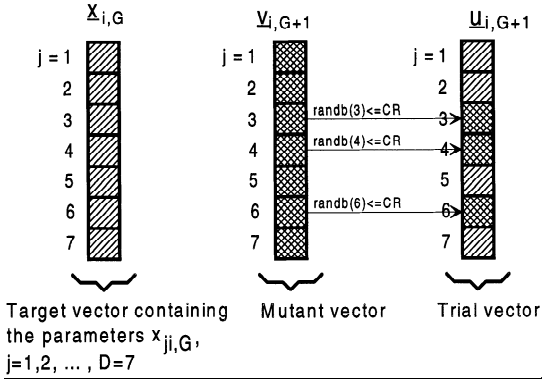

Figure 2. Illustration of the crossover process for $D = 7$ parameters.

is formed, where

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (randb(j) \leq CR) \text{ or } j = rnbr(i) \\ x_{ji,G} & \text{if } (randb(j) > CR) \text{ and } j \neq rnbr(i) \end{cases}, \quad j = 1, 2, \ldots, D. \tag{4}$$

In (4), $randb(j)$ is the jth evaluation of a uniform random number generator with outcome $\in [0, 1]$. CR is the crossover constant $\in [0, 1]$ which has to be determined by the user. $rnbr(i)$ is a randomly chosen index $\in 1, 2, ..., D$ which ensures that $u_{i,G+1}$ gets at least one parameter from $v_{i,G+1}$. Figure 2 gives an example of the crossover mechanism for 7-dimensional vectors.

Selection

To decide whether or not it should become a member of generation $G + 1$, the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$ using the greedy criterion. If vector $u_{i,G+1}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$; otherwise, the old value $x_{i,G}$ is retained.

Pseudocode

The simplicity of DE is best explained via some C-style pseudocode which is given in Figure 3.

Other variants of DE

The above scheme is not the only variant of DE which has proven to be useful. In order to classify the different variants, the notation:

$$DE/x/y/z$$

is introduced where

```
/*---------------Main loop------------------------------------------*/
while (count < gen_max)         /* Halt after gen_max generations. */
{
   for (i=0; i<NP; i++)  /*-------Start loop through population.---*/
   {
                              /********** Mutate/recombine **********/

      do a=rnd_uni()*NP; while (a==i); /* Randomly pick 3 vectors, */
      do b=rnd_uni()*NP; while (b==i || b==a);    /* all different */
      do c=rnd_uni()*NP; while (c==i || c==a || c==b);  /* from i. */
      j=rnd_uni()*D;          /* Randomly pick the first parameter. */
      for (k=1; k<=D; k++)       /* Load D parameters into trial[]. */
      {                             /* Perform D-1 binomial trials. */
         if (rnd_uni() < CR || k==D)     /* Source for trial[j] is */
         {           /* a random vector plus weighted differential */
            trial[j]=x1[c][j]+F*(x1[a][j]-x1[b][j]);      /* or... */
         }          /* trial parameter comes from target vector    */
         else trial[j]=x1[i][j];               /* x1[i][j] itself. */
         j=(j+1)%D;               /* get next parameter, modulo D. */
      }    /* Last parameter (k=D) comes from noisy random vector. */

                               /********** Evaluate/select **********/

      score=evaluate(trial); /* Evaluate trial with your function. */
      if (score<=cost[i])       /* If trial[] improves on x1[i][], */
      {                         /* move trial[] to secondary array */
         for (j=0; j<D; j++) x2[i][j]=trial[j];
         cost[i]=score;                 /* and store improved cost */
      }              /* otherwise, move x1[i][] to secondary array. */
      else for (j=0; j<D; j++) x2[i][j]=x1[i][j];
   }                 /* Mutate/recombine next primary array vector. */

         /*----------End of population loop; swap arrays----------*/

   for (i=0; i<NP; i++)                 /*  After each generation, */
   {                    /* move secondary array into primary array. */
      for (j=0; j<D; j++) x1[i][j]=x2[i][j];
   }                     /* ...or just swap pointers (not shown). */
   count++;             /* End of generation...increment counter. */


}
/*-------End of main loop---------------------------------------- */
```

Figure 3. The DE search engine in 19 lines of C-style pseudocode.

- x specifies the vector to be mutated which currently can be “rand” (a randomly chosen population vector) or “best” (the vector of lowest cost from the current population).
- y is the number of difference vectors used.
- z denotes the crossover scheme. The current variant is “bin” (Crossover due to independent binomial experiments as explained in Section 2)

Using this notation, the basic DE-strategy described in the previous chapter can be written as: DE/rand/1/bin. This is the DE-variant we used for all performance comparisons later on. Nevertheless, one highly beneficial method that deserves special mention is the method DE/best/2/bin: Price (1996), where

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r_1,G} + x_{r_2,G} - x_{r_3,G} - x_{r_4,G}). \tag{5}$$

The usage of two difference vectors seems to improve the diversity of the population if the number of population vectors NP is high enough.

3. Comparison with Other Minimization Methods

DE has already participated in the First International IEEE Competition on Evolutionary Optimization, the 1st ICEO: Price, K. and Storn, R. (1996). Among conference entries, DE proved to be the fastest evolutionary algorithm, although it did place third in speed behind two deterministic methods of limited application. Encouraged by these results, we looked for more minimization methods which claim to work effectively on real functions. The number of different minimization methods published is enormous: Schwefel (1995), so we confined ourselves to the most prominent ones.

3.1. Annealing Methods

Two annealing methods were chosen to compete with DE, the Annealed Nelder and Mead strategy (ANM): Press (1992), and Adaptive Simulated Annealing (ASA): Ingber (1993). We used readily available source code to test both annealing algorithms on a testbed which we think is challenging for global optimization methods.

The first method, ANM, is appealing because of its adaptive scheme for generating random parameter deviations. When the annealing part is switched off, a fast converging direct search method remains which is especially useful in cases where local minimization suffices. The basic control variables in ANM are T, the starting temperature, TF, the temperature reduction factor and NV, the number of random variations at a given temperature level.

The second method, ASA, claims to converge very quickly and to outperform GAs on the De Jong test suite: Ingber (1992). Although ASA provides more than a dozen control variables, it turned out that just two of them, TEMPERATURE_RATIO_SCALE (TRS) and TEMPERATURE_ANNEAL_SCALE (TAS), had significant impact on the minimization process.

Testbed #1

Our function testbed contains the De Jong test functions in a slightly modified fashion plus some additional functions which exhibit further distinctive difficulties for a global minimization algorithm. For all functions an initial parameter range, IPR, and a value to reach, VTR, were defined. At the beginning of the optimization the initial parameter values are drawn randomly from the IPR. If a minimizer gets below the VTR, the problem is assumed to be solved.

(1) First De Jong function (sphere)

$$f_1(x) = \sum_{j=1}^{3} x_j^2; \qquad \text{IPR: } x_j \in [-5.12, 5.12] \tag{6}$$

$f_1(x)$ is considered to be a very simple task for every serious minimization method. The global minimum is $f_1(0) = 0$ and the VTR was set to $1.e^{-6}$.

(2) Second De Jong function (Rosenbrock's saddle)

$$f_2(x) = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2; \quad \text{IPR: } x_j \in [-2.048, 2.048] \quad (7)$$

Although $f_2(x)$ has just two parameters, it has the reputation of being a difficult minimization problem. The global minimum is $f_2(1)=0$ and VTR=$1.e^{-6}$.

(3) Modified third De Jong function (step)

$$f_3(x) = \begin{cases} 30. + \sum_{j=1}^{5} \lfloor x_j \rfloor & \forall x_j \in IPR \\ \prod_{(x_j \notin IPR) \wedge (x_j < 0)} 30 \cdot sgn(-x_j - 5.12) \end{cases};$$

$$\text{IPR: } x_j \in [-5.12, 5.12] \quad (8)$$

The global minimum is $f_3(-5 - \varepsilon) = 0$ where $\varepsilon \in$ [0,0.12]. Again, the VTR was chosen to be $1.e^{-6}$.The step function exhibits many plateaus which pose a considerable problem for many minimization algorithms. The original step function does not contain the product of signum functions which causes the minimum to occur at minus infinity. In order to define a unique global minimum, however, the signum functions were included.

(4) Modified fourth De Jong function (quartic)

$$f_4(x) = \sum_{j=1}^{30} (j \cdot x_j^4 + \eta); \qquad \text{IPR: } x_j \in [-1.28, 1.28] \quad (9)$$

This function is designed to test the behavior of a minimization algorithm in the presence of noise. In the original De Jong function, η is a random variable produced by Gaussian noise having the distribution $N(0, 1)$. According to Ingber (1992), this function appears to be flawed as no definite global minimum exists. In response to the problem, we followed the suggestion given in Ingber (1992) and chose η to be a random variable with uniform distribution and bounded by [0,1). In contrast to the original version of De Jong's quartic function, we have also included η inside the summation instead of just adding η to the summation result. This change makes $f_4(x)$ more difficult to minimize. The functional minimum is $f_4(0) \leq 30\text{E}[\eta] = 15 = \text{VTR}$, where $\text{E}[\eta]$ is the expectation of η.

(5) Fifth De Jong function (Shekel's Foxholes)

$$f_5(x) = \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{i + \sum_{j=1}^{2} (x_j - a_{ij})^6}};$$

$$\text{IPR: } x_j \in [-65.536, 65.536] \quad (10)$$

with $a_{i1} = \{-32, -16, 0, 16, 32\}$ for $i = 0, 1, 2, 3, 4$ and $a_{i1} = a_{i \bmod 5,1}$ as well as $a_{i2} = \{-32, -16, 0, 16, 32\}$ for $i = 0, 5, 10, 15, 20$ and $a_{i2} = a_{i+k,2}$, $k = 1, 2, 3, 4$

The global minimum for this function is $f_6(-32,-32) \cong 0.998004$, the VTR was defined to be 0.998005.

(6) Corana's parabola: Ingber (1993), Corana *et al*. (1987).

$$f_6(x) = \sum_{j=1}^{4} \begin{cases} 0.15 \cdot (z_j - 0.05 \cdot sgn(z_j))^2 \cdot d_j & \text{if } |x_j - z_j| < 0.05 \\ d_j \cdot x_j^2 & \text{otherwise} \end{cases};$$

$$\text{IPR: } x_j \in [-1000, 1000] \tag{11}$$

with

$$z_j = \left\lfloor \left| \frac{x_j}{0.2} \right| + 0.49999 \right\rfloor \cdot sgn(x_j) \cdot 0.2$$

and

$$d_j = \{1, 1000, 10, 100\}$$

$f_6(x)$ defines a paraboloid whose axes are parallel to the coordinate axes. It is riddled with a set of holes that increase in depth the closer one approaches the origin. Any minimization algorithm that goes strictly downhill will almost always be captured by the holes. The global minimum here is $f_6(x) = 0$, with $|x_j| < 0.05$, $j = 1, 2, 3, 4$. The VTR was defined to be $1.e^{-6}$.

(7) Griewangk's function: Griewangk (1981).

$$f_7(x) = \sum_{j=1}^{10} \frac{x_j^2}{4000} - \prod_{j=1}^{10} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1; \text{ IPR: } x_j \in [-400, 400] \tag{12}$$

Like test function $f_6(x)$, $f_7(x)$ has many local minima so that it is very difficult to find the true minimum $f_7(0) = 0$. The VTR was defined to be $1.e^{-6}$.

(8) Zimmermann's problem: Zimmermann (1990).

Let

$$h_1(x) = 9 - x_1 - x_2, \qquad \text{IPR: } x_j \in [0, 100], \quad j = 1, 2 \tag{13}$$

$$h_2(x) = (x_1 - 3)^2 + (x_2 - 2)^2 - 16, \tag{14}$$

$$h_3(x) = x_1 \cdot x_2 - 14 \tag{15}$$

and

$$p(\delta) = 100 \cdot (1 + \delta). \tag{16}$$

Then

$$f_8(x) = \max\{h_1(x), p(h_2(x)) \cdot sgn(h_2(x)), p(h_3(x)) \cdot sgn(h_3(x)), p(-x_1) \cdot sgn(-x_1), p(-x_2) \cdot sgn(-x_2)\}. \tag{17}$$

Finding the global minimum $f_8(7, 2) = 0$ poses a special problem, because the minimum is located at the corner of the constrained region defined by $(x_1 - 3)^2 + (x_2 - 2)^2 \leq 16$ and $x_1 \cdot x_2 \leq 14$. The VTR was defined to be $1.e^{-6}$.

(9) Polynomial fitting problem by Storn and Price.

Let

$$h_4(x, z) = \sum_{j=0}^{2k} x_{j+1} \cdot z^j, k \text{ integer and} > 0, \tag{18}$$

have the coefficients x_{j+1} such that

$$h_4(x, z) \in [-1, 1] \quad \text{for} \quad z \in [-1, 1] \tag{19}$$

and

$$h_4(x,z) \geq T_{2k}(1.2) \quad \text{for} \quad z = \pm 1.2 \tag{20}$$

with $T_{2k}(z)$ being a Chebychev Polynomial of degree $2k$. The Chebychev Polynomials are defined recursively according to the difference equation $T_{n+1}(z) = 2z \cdot T_n(z) - T_{n-1}(z)$, n integer and > 0, with the initial conditions $T_0(z) = 1$ and $T_1(z) = z$. The solution to the polynomial fitting problem is, of course, $h_4(x,z) = T_{2k}(z)$, a polynomial which oscillates between -1 and 1 when its argument z is between -1 and 1. Outside this "tube" the polynomial rises steeply in direction of high positive ordinate values. The polynomial fitting problem has its roots in electronic filter design: Rabiner and Gold (1975), and challenges an optimization procedure by forcing it to find parameter values with grossly different magnitudes, something very common in technical systems. In our test suite we employed

$$T_8(z) = 1 - 32z^2 + 160z^4 - 256z^6 + 128z^8 \tag{21}$$

with

$$T_8(1.2) \cong 72.661 = \alpha \tag{22}$$

as well as

$$\begin{aligned} T_{16}(z) \;=\; & 1 - 128z^2 + 2688z^4 - 21504z^6 + 84480z^8 - \\ & -180224z^{10} + 212992z^{12} - 131072z^{14} + 32768z^{16} \end{aligned} \tag{23}$$

with

$$T_{16}(1.2) \cong 10558.145 = \alpha \tag{24}$$

and used the weighted sum of squared errors in order to transform the above constraint satisfaction problem into a cost function

$$\begin{aligned} f_9(x) \;=\; & \sum_{n=0}^{N} sgn\left(-1 + h_4\left(x, \frac{N}{n}\right)\right) \cdot \left(-1 + h_4\left(x, \frac{N}{n}\right)\right)^2 \\ & + \sum_{n=0}^{N} sgn\left(-1 - h_4\left(x, \frac{N}{n}\right)\right) \cdot \left(-1 - h_4\left(x, \frac{N}{n}\right)\right)^2 \\ & + sgn(\alpha - h_4(x, 1.2)) \cdot (\alpha - h_4(x, 1.2))^2 \\ & + sgn(\alpha - h_4(x, -1.2)) \cdot (\alpha - h_4(x, -1.2))^2. \end{aligned} \tag{25}$$

The initial conditions for the T_8 problem ($k = 4$) were $N = 60$ and IPR $[-100, 100]$, for the T_{16} problem ($k = 8$) they were $N = 100$ and IPR $[-1000, 1000]$. The VTR was defined to be $1.e^{-6}$.

Test Results

We experimented with each of the four algorithms to find the control settings which provided fastest and smoothest convergence. Table 1 contains our choice of control variable settings for each minimization algorithm and each test function along with the averaged number of function evaluations (nfe) which were required to find the global minimum.

Table 1. Averaged number of function evaluations (nfe) required for finding the global minimum. A hyphen indicates misconvergence and n.a. parentheses indicate that not all test runs were able to locate the global minimum.

$f_i(x)$	ANM				ASA			DE/rand/1/bin			
i	T	TF	NV	nfe	TRS	TAS	nfe	NP	F	CR	nfe
1	0	n.a.	1	95	10^{-5}	10	397	5	0.9	0.1	406
2	0	n.a.	1	106	10^{-5}	10000	11,275	10	0.9	0.9	654
3	300	0.99	20	90,258	10^{-7}	100	354	10	0.9	0	849
4	300	0.98	30	-	10^{-5}	100	4,812	10	0.9	0	859
5	3000	0.995	50	-	10^{-5}	100	1,379	15	0.9	0	695
6	$5{\cdot}10^6$	0.995	100	-	10^{-5}	100	3,581	10	0.5	0	841
7	10	0.99	50	-	10^{-5}	0.1	-	25	0.5	0.2	12,752
8	5	0.95	5	2,116	10^{-6}	300	11,864	10	0.9	0.9	925
$9(k=4)$	100	0.95	40	(391,373)	10^{-6}	1000	-	60	0.6	1	15,771
$9(k=8)$	$5{\cdot}10^4$	0.995	150	-	10^{-8}	700	-	100	0.6	1	93,650

If the corresponding field for the number of function evaluations contains a hyphen, the global minimum could not be found. If the number is enclosed in parentheses, not all of the test runs provided the global minimum. We executed 20 test runs with randomly chosen initial parameter vectors for each test function and each minimization.

The results in Table 1 clearly show that DE was the only strategy that could find all global minima of the test suite. Except for the test functions 1, 2 and 3 DE found the minimum in the least number of function evaluations. It is also noteworthy that for $f_9(x)$ DE found the global minimum even though the final parameters lie well outside the IPR.

3.2. Evolutionary Algorithms

Evolution Strategies (ESs) as well as Genetic Algorithms (GAs) belong to the broad class of evolutionary algorithms: Schwefel (1995), which have achieved impressive results in continuous parameter optimization problems: Voigt (1995). Two particularly effective approaches are the Breeder Genetic Algorithm (BGA): Muehlenbein (1993) and the Evolutionary Algorithm with Soft Genetic Operators (EASY): Voigt (1995). Both algorithms have performed well on a testbed with highly multimodal test functions of medium to high dimensionality. Unlike the annealing algorithms, source code was not available so we had to rely on the reported results and accept the settings of the testbed. In several instances, the VTRs seem not to be challenging enough to test the local optimization capabilities of the minimizers. As many symmetries are present, the main difficulty of these test functions lies in their dimensionality.

Testbed #2

(1) Hyper-Ellipsoid

$$f_{11}(x) = \sum_{j=1}^{D} j^2 \cdot x_j^2; \qquad \text{IPR: } x_j \in [-1, 1] \tag{26}$$

The global minimum is $f_{11}(0) = 0$ and the VTR was set to $1.e^{-10}$.

(2) Katsuura's Function

$$f_{12}(x) = \prod_{j=1}^{D} \left(1 + j \cdot \sum_{k=0}^{\beta} \left| 2^k \cdot x_j - nint(2^k \cdot x_j) \right| \cdot 2^{-k} \right);$$
$$\text{IPR: } x_j \in [-1000, 1000] \tag{27}$$

Here nint() denotes the function which finds the nearest integer and the constant β is set to 32. The global minimum is $f_{12}(0) = 1$ and the VTR was set to 1.05.

(3) Rastrigin's Function

$$f_{13}(x) = D \cdot 10 + \sum_{j=1}^{D} (x_j^2 - 10 \cdot \cos(2\pi \cdot x_j)); \text{ IPR: } x_j \in [-600, 600] \tag{28}$$

The global minimum is $f_{13}(0) = 0$ and the VTR was set to 0.9.

(4) Griewangk's function

$$f_{14}(x) = \sum_{j=1}^{D} \frac{x_j^2}{4000} - \prod_{j=1}^{D} \cos\left(\frac{x_j}{\sqrt{j}} \right) + 1; \text{ IPR: } x_j \in [-600, 600] \tag{29}$$

Eq. (29) is a generalization of (12) with the same global minimum $f_{14}(0) = 0$. The VTR here was defined to be $1.e^{-3}$.

(5) Ackley's function

$$f_{15}(x) = -20 \cdot \exp\left(-0.02 \cdot \sqrt{D^{-1} \cdot \sum_{j=1}^{D} x_j^2} \right)$$
$$- \exp\left(D^{-1} \cdot \sum_{j=1}^{D} \cos(2\pi \cdot x_j) \right) + 20 + \exp(1);$$
$$\text{IPR: } x_j \in [-30, 30] \tag{30}$$

The global minimum is $f_{15}(0) = 0$ and the VTR is set to $1.e^{-3}$.

As already reported in Price (1996), DE/best/2/bin performs very well on this testbed. The results for BGA, EASY and DE/rand/bin/1 are summarized in Table 2.

Test Results

DE performed favorably against both BGA and EASY (EASY-1: Voigt (1992), to be more precise) and needed the least number of function evaluations in 8 of 10 cases. This result is quite remarkable considering the simplicity of the DE-heuristic compared to the intricacies of the two contenders. The DE-results were computed

Table 2. Averaged number of function evaluations (nfe) required to find the global minimum. NA stands for "not available".

$\mathbf{f}_i(x)$	D	nfe			DE-Settings		
i		BGA	EASY	DE/rand/1/bin	NP	F	CR
11	30	NA	27,111	16,907	20	0.5	0.1
11	100	NA	104,520	56,145	20	0.5	0.1
12	10	NA	9,626	4,269	15	0.5	0.1
12	30	NA	39,333	12,859	15	0.5	0.1
13	20	3,608	6,098	12,971	25	0.5	0
13	100	25,040	45,118	73,620	25	0.5	0
14	20	66,000	26,700	8,691	20	0.5	0.1
14	100	361,722	77,250	31,796	20	0.5	0.1
15	30	19,420	13,997	12,481	20	0.5	0.1
15	100	53,860	57,628	36,801	20	0.5	0.1

by averaging 20 minimization runs for each test case. Each run found the global minimum.

3.3. STOCHASTIC DIFFERENTIAL EQUATIONS

The method of stochastic differential equations (SDE): Aluffi-Pentini *et al*. (1985) requires a cost function to be at least partially differentiable. This is a condition which usually holds true for practical problems. The basic idea is to perturb a gradient search by a stochastic process while the perturbation gradually decreases over time. The method, which shares many similiarities with Simulated Annealing, performs reasonably well on many problems, making it another contender for DE. Again, no source code was available, so we used the testbed and the settings given in Aluffi-Pentini *et al*. (1985). Unfortunately, neither IPR nor VTR is provided in the above reference, so we had to make some assumptions regarding IPR and VTR.

Testbed #3

For most test functions, the VTR was defined to be the actual global minimum with a relative accuracy of $1.e^{-6}$. For those functions which have a global minimum value of 0, the VTR itself was set to $1.e^{-6}$. The IPR was always $x_j \in [-10, 10]$ except for $f_{30}(x)$ where the IPR was $x_j \in [-e^{+4}, e^{+4}]$.

(1) Goldstein's function

$$f_{16}(x) = x^6 - 15x^4 + 27x^2 + 250. \tag{31}$$

The global minimum is $f_{16}(+/-3) = 7$.

(2) Penalized Shubert Function

$$f_{17}(x) = g_1(x) + u(x, 10, 100, 2); \tag{32}$$

where

$$g_1(x) = \sum_{i=1}^{5} i \cdot \cos((i+1) \cdot x + i) \tag{33}$$

is the Shubert function and the penalization function $u(z, a, k, m)$ is defined by

$$u(z, a, k, m) = \begin{cases} k(z-a)^m, & z > a \\ 0, & -a \le z \le a \\ k(-z-a)^m, & z < -a \end{cases} . \tag{34}$$

The global minimum is $f_{17}(-7.70831) = f_{17}(-1.42513) = f_{17}(-4.85805) = -12.8708855$.

(3) Two-dimensional Penalized Shubert Function

$$f_{18}(x) = g_1(x_1) \cdot g_1(x_2) + u(x_1, 10, 100, 2) + u(x_2, 10, 100, 2). \tag{35}$$

There are 18 global minima with value -186.7309088 in the region bounded by the IPR.

(4) Two-dimensional modified and penalized Shubert Function

$$f_{19}(x) = f_{18}(x) + \beta \cdot [(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2]. \tag{36}$$

The global minimum is $f_{19}(-1.42513, -0.80032) = -186.7309088$.

(5) Six-hump Camel Function

$$\begin{aligned} f_{20}(x) \;=\; & \left(4 - 2.1 \cdot x_1^2 + \frac{1}{3} \cdot x_1^4\right) + x_1 \cdot x_2 \\ & +(-4 + 4 \cdot x_2^2) \cdot x_2^2. \end{aligned} \tag{37}$$

The global minimum is $f_{20}(-0.0898, 0.7126) = f_{20}(0.0898, -0.7126) = -1.0316285$.

(6) The next function is defined as

$$f_{21}(x) = g_2(x) + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \tag{38}$$

with

$$\begin{aligned} g_2(x) \;=\; & \frac{\pi}{D} \left\{ 10 \cdot \sin^2\left(\pi + \frac{\pi}{4} \cdot (x_1 - 1)\right) \right. \\ & + \sum_{i=1}^{D-1} 0.125 \cdot (x_i - 1)^2 \cdot \left[1 + 10 \cdot \sin^2\left(\pi + \frac{\pi}{4} \cdot (x_{i+1} - 1)\right)\right] \\ & \left. +0.125 \cdot (x_D - 1)^2 \right\}; \end{aligned} \tag{39}$$

The global minimum is $f_{21}(1) = 0$.

(7) The next function is defined as

$$f_{22}(x) = g_3(x) + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \tag{40}$$

with

$$g_3(x) \;=\; \frac{\pi}{D} \left\{ 10 \cdot \sin^2(\pi \cdot x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \cdot \right.$$

$$\cdot[1 + 10 \cdot \sin^2(\pi \cdot x_{i+1})] + (x_D - 1)^2\Big\}. \tag{41}$$

The global minimum is $f_{22}(1) = 0$.

(8) The next function is defined as

$$f_{23}(x) = g_4(x) + \sum_{i=1}^{D} u(x_i, 10, 100, 4) \tag{42}$$

with

$$g_4(x) = 0.1\Big\{\sin^2(3\pi \cdot x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \cdot [1 + \sin^2(3\pi \cdot x_{i+1})] + (x_D - 1)^2 \cdot [1 + \sin^2(2\pi \cdot x_D)]\Big\}. \tag{43}$$

The global minimum is $f_{23}(1) = 0$.

(9) The next function is defined as

$$f_{24}(x) = g_4(x) + \sum_{i=1}^{D} u(x_i, 5, 100, 4) \tag{44}$$

The global minimum is $f_{24}(1) = 0$.

(10) The next function is defined as

$$f_{25}(x) = 0.25 \cdot x^4 - 0.5 \cdot x^2 + 0.1 \cdot x \tag{45}$$

The global minimum is $f_{25}(-1.0466805696) = -0.3523861$.

(11) The next function is defined as

$$f_{26}(x) = 0.25 \cdot x_1^4 - 0.5 \cdot x_1^2 + 0.1 \cdot x_1 + 0.5 \cdot x_2^2 \tag{46}$$

The global minimum is $f_{26}(-1.0466805696, 0) = -0.3523861$.

(12) The next function is defined as

$$f_{27}(x) = 0.5 \cdot x_1^2 + 0.5 \cdot [1 - \cos(2x_1)] + x_2^2 \tag{47}$$

The global minimum is $f_{27}(0) = 0$.

(13) The next function is defined as

$$f_{28}(x) = 10^n x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^m (x_1^2 + x_2^2)^4; n > 0, m < 0. \tag{48}$$

The global minimum is

$$f_{28}(0, +/- 1.38695228) = -0.4074616 (n = -m = 1).$$
$$f_{28}(0, +/- 2.60890651) = -18.0586967 (n = -m = 2)$$
$$f_{28}(0, +/- 4.70173979) = -227.7657500 (n = -m = 3)$$
$$f_{28}(0, +/- 8.39400578) = -2429.4147670 (n = -m = 4)$$
$$f_{28}(0, +/- 14.94511228) = -24776.5183423 (n = -m = 5)$$
$$f_{28}(0, +/- 26.58677673) = -249293.0182630 (n = -m = 6)$$

(14) The next function is defined as

$$f_{29}(x) = \sqrt[4]{\sum_{i=1}^{5} i \cdot x_i^2} \tag{49}$$

The global minimum is $f_{29}(0) = 0$.

(15) The next function is defined as

$$f_{30}(x) = -F(x) + u(x_1, 10^4, 100, 2) + u(x_2, 10^4, 100, 2),$$
$$\text{IPR: } x_j \in [-\mathrm{e}^{+4}, \mathrm{e}^{+4}] \tag{50}$$

with

$$F(x) = \prod_{i=1}^{14} \left[\frac{\Phi(z_i - x_1)}{x_2}\right]^{1-\delta_i} \cdot \left[1 - \frac{\Phi(z_i - x_1)}{x_2}\right]^{\delta_i} \tag{51}$$

and

$$\Phi(z) = 0.5\left(1 + erf\left(\frac{z}{\sqrt{2}}\right)\right) \tag{52}$$

and erf(z) denoting the error function. The z_i are taken from

$\{1219, 1371, 1377, 1144, 1201, 1225, 1244, 1254, 1304, 1328,$
$1351, 1356, 1370, 1390\}$

in natural order. Analogously the δ_i are taken from

$\{0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$.

The global minimum is $f_{30}(1523.2, 277.5) = -0.000888085$.

The necessary number of function evaluations (nfe) for both SDE and DE are shown in Table 3.

Test Results

Table 3 shows in all test cases that DE exhibits superior performance when compared to SDE's reported best results. DE's mean number of required function evaluations was computed by averaging the results of 1000 trial runs for each function. None of the trial runs failed to find the global minimum. It is also noteworthy that DE's control variable settings could remain the same for most of the functions, which is an indication of DE's robustness. Only f_{19} and f_{30} called for a different setting to achieve regular convergence. In testcase $f_{28}(x)$ with $n = 5, 6$ the optimum parameter values lie outside the IPR, yet DE had no difficulty finding them.

4. Choice of DE's Control Variables

It is interesting to note that DE's control variables, NP, F and CR, are not difficult to choose in order to obtain good results. According to our experience a reasonable choice for NP is between 5∗D and 10∗D but NP must be at least 4 to ensure that DE will have enough mutually different vectors with which to work. As for F, F = 0.5 is usually a good initial choice. If the population converges prematurely, then F and/or NP should be increased. Values of F smaller than 0.4, like those greater than 1, are only occasionally effective. A good first choice for CR is 0.1, but since a large CR often speeds convergence, to first try CR = 0.9 or CR = 1.0 is appropriate in order to see if a quick solution is possible. For fastest convergence, it is best to pick the IPR such that it covers the region of the suspected global optimum, although this choice doesn't seem to be mandatory. These rules of thumb for DE's

Table 3. Averaged number of function evaluations (nfe) required to find the global minimum.

$f_i(x)$	nfe		DE-Settings		
i	SDE	DE/rand/1/bin	NP	F	CR
16	3,184	503	20	0.5	0
17	26,893	499	20	0.5	0
18	241,215	3,137	20	0.5	0
19, β=0.5	8,755	4,854	40	1.0	0
19, β=1.0	97,761	4,428	40	1.0	0
20	5,393	927	20	0.5	0
21, D=2	84,782	722	20	0.5	0
21, D=3	19,041	1,073	20	0.5	0
21, D=4	18,942	1,424	20	0.5	0
22, D=5	18,433	2,084	20	0.5	0
22, D=8	136,061	3,347	20	0.5	0
22, D=10	49,701	4,165	20	0.5	0
23, D=2	9,492	715	20	0.5	0
23, D=3	19,114	1,093	20	0.5	0
23, D=4	35,139	1,499	20	0.5	0
24, D=5	53,398	1,882	20	0.5	0
24, D=6	15,534	2,295	20	0.5	0
24, D=7	16,542	2,701	20	0.5	0
25	6,751	273	20	0.5	0
26	3,402	650	20	0.5	0
27	10,286	621	20	0.5	0
28, n= −m=1	4,791	907	20	0.5	0
28, n= −m=2	3,037	812	20	0.5	0
28, n= −m=3	5,028	778	20	0.5	0
28, n= −m=4	14,710	754	20	0.5	0
28, n= −m=5	51,285	751	20	0.5	0
28, n= −m=6	17,610	761	20	0.5	0
29	15,102	7,053	20	0.5	0
30	48,802	1,266	30	0.5	1.0

control variables render DE fairly easy to work with which is one of DE's major assets.

5. Conclusion and Final Thoughts

The Differential Evolution method (DE) for minimizing continuous space functions has been introduced and compared to Adaptive Simulated Annealing (ASA), the Annealed Nelder and Mead approach (ANM), the Breeder Genetic Algorithm (BGA), The EASY Evolution Strategy and the method of Stochastic Differential Equations (SDE). In most instances, DE outperformed all of the above minimiza-

tions approaches in terms of required number of function evaluations necessary to locate a global minimum of the test functions. This excellent result is especially intriguing since DE is a very simple and straightforward strategy. Indeed DE's main search engine can be written in less than 30 lines of C-code. DE is also very easy to use as it requires only a few robust control variables which can be drawn from a well-defined numerical interval. Although DE has shown promising results, it is still in its infancy and can most probably be improved. We think that DE's method of self-organization is remarkable and should be investigated more deeply. Further research should include a mathematical convergence proof like the one that exists for Simulated Annealing. Practical experience shows that DE's vector generation scheme leads to a fast increase of population vector distances if the objective function surface is flat. This "divergence property" prevents DE from advancing too slowly in shallow regions of the objective function surface and allows for quick progress after the population has traveled through a narrow valley. If the vector population approaches the final minimum, the vector distances decrease due to selection, allowing the population to converge reasonably fast. Despite these insights derived from experimentation, a theoretically sound analysis to determine why DE converges so well would be of great interest.

Little is known about DE's scaling property and behavior in real-world applications. The most complex real-world applications solved with DE so far are the design of a recursive digital filter with 18 parameters and with multiple constraints and objectives: Storn (1996a), the design of a 60-parameter linear phase finite impulse response filter: Storn (1996c), and a communications control problem with 30 pararmeters: Ziny (1995). Many problems, however, are much larger in scale and DE's behavior in such cases is still unknown.

Whether or not the combination of DE with other optimization approaches is of use, also has yet to be answered. Finally, it is important for quickly solving practical applications that we gain more knowledge on how to choose the control variables for DE for a particular type of problem.

References

1. Aluffi-Pentini, F., Parisi, V. and Zirilli, F. (1985), Global Optimization and Stochastic Differential Equations, *Journal of Optimization Theory and Applications* 47 (1), 1–16.
2. Brayton, H., Hachtel, G. and Sangiovanni-Vincentelli, A. (1981), A Survey of Optimization Techniques for Integrated Circuit Design, *Proceedings of the IEEE* 69, pp. 1334–1362.
3. Bunday, B.D. and Garside G.R. (1987), *Optimisation Methods in Pascal*, Edward Arnold Publishers.
4. Corana, A., Marchesi, M., Martini, C. and Ridella, S. (1987), *Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing Algorithm"*, ACM Transactions on Mathematical Software, March 1987, pp. 272–280.
5. Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley.
6. Griewangk, A.O. (1981), Generalized Descent for Global Optimization, *JOTA* 34, 11–39.

7. Ingber, L. and Rosen, B. (1992), Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, *J. of Mathematical and Computer Modeling* 16 (11), 87–100.
8. Ingber, L. (1993), Simulated Annealing: Practice Versus Theory, *J. of Mathematical and Computer Modeling* 18 (11), 29–57.
9. Lueder, E. (1990), Optimization of Circuits with a Large Number of Parameters, *Archiv fuer Elektronik und Uebertragungstechnik* 44 (2), 131–138.
10. Muehlenbein, H. and Schlierkamp-Vosen (1993), Predictive Models for the Breeder Genetic Algorithm, I. Continuous Parameter Optimizations, *Evolutionary Computation* 1 (1), 25–49.
11. Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992), *Numerical Recipes in C*, Cambridge University Press.
12. Price, K. (1994), Genetic Annealing, *Dr. Dobb's Journal*, Oct. 1994, 127–132.
13. Price, K. and Storn, R. (1996), Minimizing the Real Functions of the ICEC'96 contest by Differential Evolution, *IEEE International Conference on Evolutionary Computation* (ICEC'96), may 1996, pp. 842–844 .
14. Price, K. (1996), *Differential Evolution: A Fast and Simple Numerical Optimizer*, NAFIPS'96, pp. 524–527.
15. Rabiner, L.R. and Gold, B. (1975), *Theory and Applications of Digital Signal Processing*, Prentice- Hall, Englewood Cliffs, N.J..
16. Rechenberg, I. (1973), *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.
17. Schwefel, H.P. (1995), *Evolution and Optimum Seeking*, John Wiley.
18. Storn, R. (1995), Constrained Optimization, *Dr. Dobb's Journal*, May 1995, 119–123.
19. Storn, R. (1996a), Differential Evolution Design of an IIR-Filter, *IEEE International Conference on Evolutionary Computation* (ICEC'96), May 1996, pp. 268–273.
20. Storn, R. (1996b), *On the Usage of Differential Evolution for Function Optimization*, NAFIPS'96, pp. 519–523.
21. Storn, R. (1996c), Design of an FIR-filter with Differential Evolution, private communication, 1996.
22. Voigt, H.-M. (1995), *Soft Genetic Operators in Evolutionary Computation, Evolution and Biocomputation*, Lecture Notes in Computer Science 899, Springer, Berlin, pp. 123–141.
23. Zimmermann, W. (1990), *Operations Research*, Oldenbourg.
24. Ziny, F., Optimization of routing control with Differential Evolution, private communication, 1995.