Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces

by **Rainer Storn**

International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704,
storn@icsi.berkeley.edu, rainer.storn@zfe.siemens.de

and **Kenneth Price**

836 Owl Circle, Vacaville, CA 95687, kprice@solano.community.net

Keywords: stochastic optimization, nonlinear optimization, constrained optimization, numerical function minimization, global optimization

Running Head and Abstract

A new heuristic approach for minimizing possibly nonlinear and non differentiable continuous space functions is presented. By means of an extensive testbed, which includes the De Jong functions, it is demonstrated that the new method converges faster and with more certainty than both Adaptive Simulated Annealing and the Annealed Nelder&Mead approach. The new method requires few control variables, is robust, easy to use, and lends itself very well to parallel computation.

Mailing address: (until september, 30, 1996)

Dr. Rainer Storn

International Computer Science Institute,

1947 Center Street, Berkeley, CA 94704,

E-mail: storn@icsi.berkeley.edu

Fax: 510-643-7684

Phone: 510-642-4274 (extension 192)

Mailing address: (from october 1, 1996 on)

Dr. Rainer Storn

Siemens AG, ZFE T SN 2

Otto-Hahn-Ring 6

D-81739 Muenchen

Germany,

E-mail: rainer.storn@zfe.siemens.de

Fax: 01149-89-636-44577

Phone: 01149-89-636-40502

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 alii (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 regard optimization methods that use the objective function. In most cases, the objective function defines the optimization problem as a minimization task. To this end, the following investigation is restricted to minimization problems.

When the objective function is nonlinear and non-differentiable, direct search approaches are the methods of choice. The best known of these are the algorithms by Nelder&Mead: Bunday et alii (1987), by Hooke&Jeeves: Bunday et alii (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 be made whether or not to accept the newly derived parameters. All 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 objective 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 alii (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 an uphill move decreases. In the long run, this leads to the greedy criterion. While all direct search methods lend themselves to annealing, it has mostly 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&Mead: Press et alii (1992) and genetic algorithms: Ingber (1993), Price (1994).

Users generally demand that a practical optimization technique should fulfill three requirements. First, the method should find the true global minimum, regardless of the initial system parameter values. Second, convergence should be fast. Third, the program should have a minimum of control parameters so that it will be easy to use. In our search for a fast and easy to use "sure fire" technique, we developed a method which is not only simple, but also performs well on a wide variety of test problems. It is inherently parallel and hence lends itself to computation via a network of computers or processors. The basic strategy employs the difference of two randomly selected parameter vectors as the source of random variations for a third parameter vector. In the following, we present a more rigorous description of the new optimization method which we call Differential Evolution.

Problem Formulation

Consider a system with the real-valued properties

$$g_m;\ m = 0, 1, 2, \ldots , P-1 \qquad (1)$$

which constitute the objectives of the system to be optimized.

Additionally, there may be real-valued constraints

$$g_m;\ m = P, P+1, \ldots , P+C-1 \qquad (2)$$

which describe properties of the system that need not be optimized but neither shall be violated. For example, one may wish to design a mobile phone with the dual objectives of maximizing the transmission power g_1 and minimizing the noise g_2 of the audio amplifier while simultaneously keeping the battery life g_3 above a certain threshold. The properties g_1 and g_2 represent objectives to be optimized whereas g_3 is a constraint. Let all properties of the system be dependent on the real-valued parameters

$$x_j;\ j = 0, 1, 2, \ldots , D-1. \qquad (3)$$

In the case of the mobile phone the parameters could be resistor and capacitor values. For most technical systems realizability requires

$$x_j \in [x_{jl}, x_{jh}], \qquad (4)$$

where x_{jl} is a lower bound on x_j and x_{jh} is the upper bound. Usually, bounds on the x_j will be incorporated into the collection g_m, $m \geq P$, of constraints. Optimization of the system means to vary the D-dimensional parameter vector

$$\underline{x} = (x_0, x_1, \ldots , x_{D-1})^T \qquad (5)$$

until the properties g_m are optimized and the constraints g_m, $m \geq P$, are met. An optimization task can always be reformulated as the minimization problem

$$\min f_m(\underline{x}) \qquad (6)$$

where $f_m(\underline{x})$ represents the function by which the property g_m is calculated and its optimization or constraint preservation is represented as the minimization of $f_m(\underline{x})$. All functions $f_m(\underline{x})$ can be combined into a single objective function $H(\underline{x})$: Lueder (1990), Moebus (1990), which usually is computed either via the weighted sum

$$H(\underline{x}) = \sum_{m=0}^{P+C-1} w_m \cdot f_m(\underline{x}) \qquad (7)$$

or via $$H(\underline{x}) = \max\left(w_m \cdot f_m(\underline{x})\right) \qquad (8)$$

with $$w_m > 0. \qquad (9)$$

The weighting factors w_m are used to define the importance associated with the different objectives and constraints as well as to normalize different physical units. The optimization task can now be restated as

$$\min H(\underline{x}) \qquad (10)$$

The min-max formulation (8) and (10) guarantees that all local minima, the Pareto critical points, including the possibly multiple global minima, the Pareto points, can at least theoretically be found: Lueder (1990), Moebus (1990). For the objective function (7) this is true only if the region of realizability of $\underline{x}$ is convex:

Brayton et alii (1981), which does not apply in most technical problems.

The Method of Differential Evolution

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

$$\underline{x}_{i,G},\ i = 0, 1, 2, \ldots, NP-1. \qquad (11)$$

as a population for each generation G. NP does not change during the minimization process. The initial population is chosen randomly if nothing is known about the system. 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 is often generated by adding normally distributed random deviations to the nominal solution $\underline{x}_{nom,0}$. The crucial idea behind DE is a scheme for generating trial parameter vectors. DE generates new parameter vectors by adding a weighted difference vector between two population members to a third member. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector will replace the vector with which it was compared in the following generation. The comparison vector can but need not be part of the generation process mentioned above. In addition the best parameter vector $\underline{x}_{best,G}$ is evaluated for every generation G in order to keep track of the progress that is made during the minimization process.

Extracting distance and direction information from the population to generate random deviations results in an adaptive scheme with excellent convergence properties. Several variants of DE have been tried, the two most promising of which are subsequently presented in greater detail.

Scheme DE1

The first variant of DE works as follows: for each vector $\underline{x}_{i,G}$, i = 0,1,2,...,NP-1, a trial vector $\underline{v}$ is generated according to

$$\underline{v} = \underline{x}_{r_1,G} + F \cdot (\underline{x}_{r_2,G} - \underline{x}_{r_3,G}), \qquad (12)$$

with $r_1, r_2, r_3 \in [0, NP-1]$, integer and mutually different, and F > 0. (13)

The integers r_1, r_2 and r_3 are chosen randomly from the interval [0, NP-1] and are different from the running index i. F is a real and constant factor which controls the amplification of the differential variation $(\underline{x}_{r_2,G} - \underline{x}_{r_3,G})$. Fig. 1 shows a two-dimensional example that illustrates the different vectors that are used in DE1.

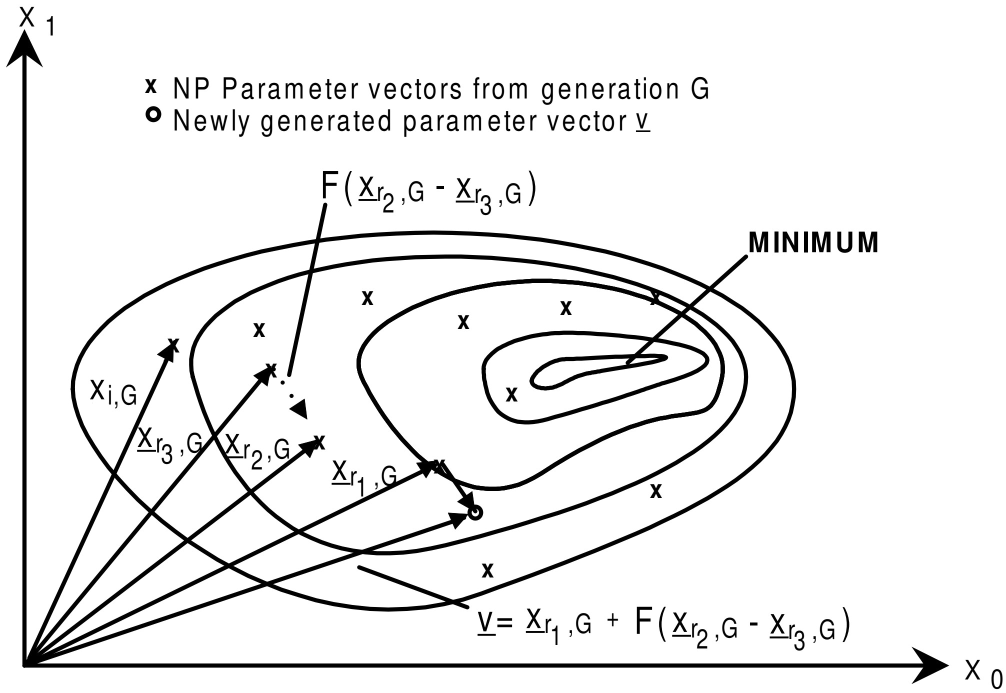

Fig.1: Two-dimensional example of an objective function showing its contour lines and the process for generating $\underline{v}$ in scheme DE1. The weighted difference vector of two arbitrarily chosen vectors is added to a third vector to yield the vector $\underline{v}$.

In order to increase the diversity of the parameter vectors, the vector

$$\underline{u} = (u_0, u_1, \ldots, u_{D-1})^T \tag{14}$$

with

$$u_j = \begin{cases} v_j & for \quad j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ (x_{i,G})_j & for \quad all \quad other \quad j \in [0, D-1] \end{cases} \tag{15}$$

is formed where the acute brackets $\langle \ \rangle_D$ denote the modulo function with modulus D.

Eqs. (14) and (15) yield a certain sequence of the vector elements of $\underline{u}$ to be identical to the elements of $\underline{v}$, the other elements of $\underline{u}$ acquire the original values of $\underline{x}_{i,G}$. Choosing a subgroup of parameters for mutation is similiar to a process known as crossover in GAs or ESs. This idea is illustrated in Fig. 2 for D=7, n=2 and L=3. The starting index n in (15) is a randomly chosen integer from the interval [0, D-1]. The integer L, which denotes the number of parameters that are going to be exchanged, is drawn from the interval [1, D]. The algorithm which determines L works according to the following lines of pseudo code where rand() is supposed to generate a random number $\in$ [0,1):

```
L = 0;
do {
    L = L + 1;
}while(rand()< CR)  AND  (L < D));
```

Hence the probability $Pr(L>=\nu) = (CR)^{\nu-1}$, $\nu > 0$. CR $\in$ [0,1] is the crossover probability and constitutes a control variable for the DE1-scheme. The random decisions for both n and L are made anew for each trial vector $\underline{v}$.

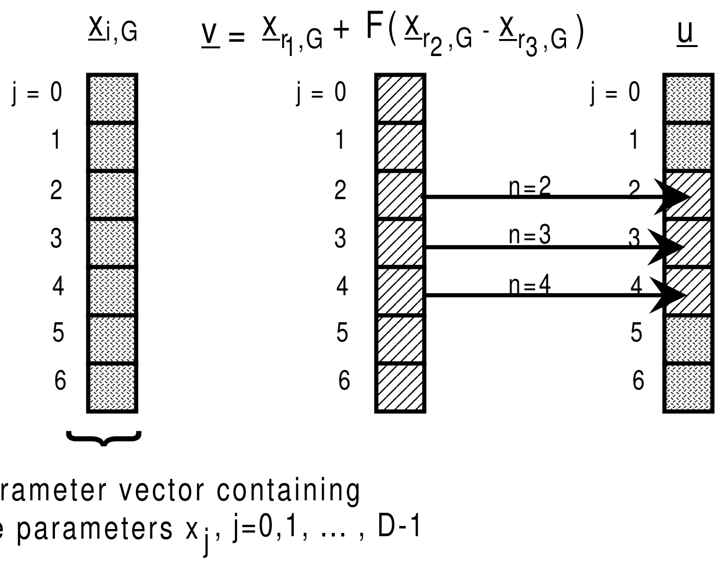

Fig. 2: Illustration of the crossover process for D=7, n=2 and L=3.

In order to decide whether the new vector $\underline{u}$ shall become a population member of generation G+1, it will be compared to $\underline{x}_{i,G}$. If vector $\underline{u}$ yields a smaller objective function value than $\underline{x}_{i,G}$, $\underline{x}_{i,G+1}$ is set to $\underline{u}$, otherwise the old value $\underline{x}_{i,G}$ is retained.

Scheme DE2

Basically, scheme DE2 works the same way as DE1 but generates the vector $\underline{v}$ according to

$$\underline{v} = \underline{x}_{i,G} + \lambda \cdot (\underline{x}_{best,G} - \underline{x}_{i,G}) + F \cdot (\underline{x}_{r_2,G} - \underline{x}_{r_3,G}) \,, \tag{16}$$

introducing an additional control variable λ. The idea behind λ is to provide a means to enhance the greediness of the scheme by incorporating the current best vector $\underline{x}_{best,G}$. This feature can be useful for objective functions where the global minimum is relatively easy to find. Fig. 3 illustrates the vector-generation process defined by (16). The construction of $\underline{u}$ from $\underline{v}$ and $\underline{x}_{i,G}$ as well as the decision process are identical to DE1.

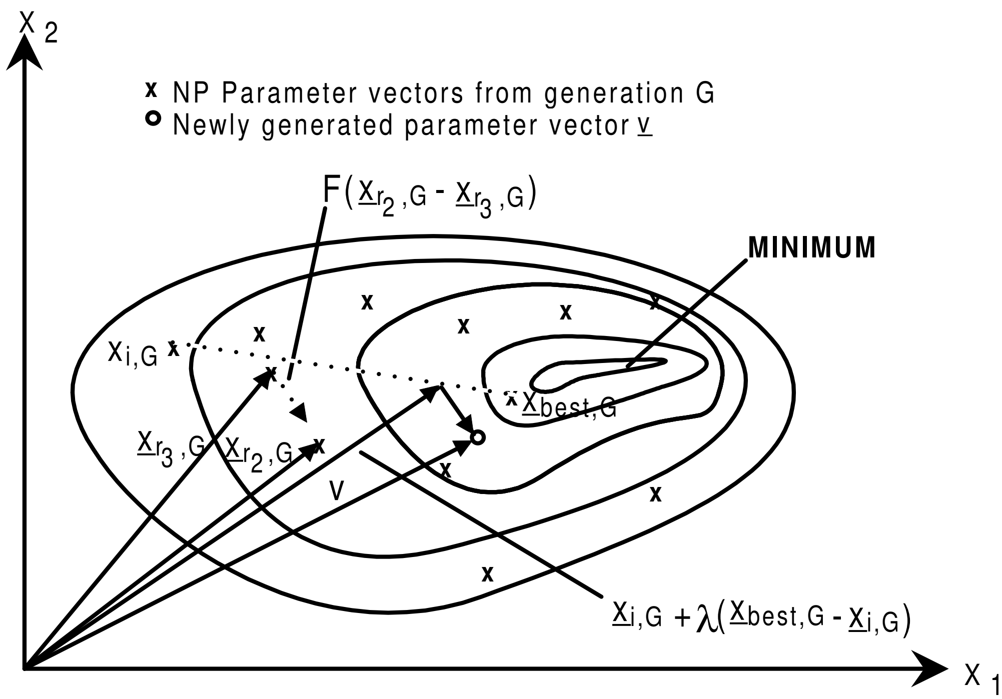

Fig.3: Two dimensional example of an objective function showing its contour lines and the process for generating $\underline{v}$ in scheme DE2.

Competing minimization methods

In order to compare the DE method with other global minimization strategies, we looked for approaches where the source code is readily available, which claim to work effectively on real functions, which require only moderate expertise for their operation, as is the case for DE itself, and which are capable of coping with nonlinear and non-differentiable functions. Two methods were chosen to compete with DE. The first was the annealed version of the Nelder&Mead strategy (ANM): Press (1992), which 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 of interest was Adaptive Simulated Annealing (ASA): Ingber (1993), which 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.

The results in Ingber (1992) were a major reason not to include GAs in the comparison. A further disadvantage of GAs is the amount of expertise that is required to operate them, the same holds true for ESs.

During our research we also wrote an annealed version of the Hooke&Jeeves method: Bunday (1987), and tested two Monte Carlo methods: Storn (1995), one of which used NP parallel vectors and the differential mutation scheme of DE. Although these approaches all worked, they quickly turned out not to be competitive.

The Testbed

Our function testbed contains the De Jong test functions as presented in Ingber (1992) plus some additional functions which present further distinctive difficulties for a global minimizer. Except for the last three, all functions are unconstrained and have a single objective with weight $w_m = w_0 = 1$ according to eqs. (7) and (8):

1) First De Jong function (sphere)

$$f_1(\underline{x}) = \sum_{j=0}^{2} x_j^2; \qquad x_j \in [-5.12, 5.12] \qquad (17)$$

$f_1(\underline{x})$ is considered to be a very simple task for every serious minimization method. The minimum is $f_1(\underline{0}) = 0$.

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

$$f_2(\underline{x}) = 100 \cdot (x_0^2 - x_1)^2 + (1 - x_0)^2; \qquad x_j \in [-2.048, 2.048] \qquad (18)$$

Although $f_2(\underline{x})$ has just two parameters, it has the reputation of being a difficult minimization problem. The minimum is $f_2(\underline{1})=0$.

3) Third De Jong function (step)

$$f_3(\underline{x}) = 30. + \sum_{j=0}^{4} \lfloor x_j \rfloor; \qquad x_j \in [-5.12, 5.12] \qquad (19)$$

For $f_3(\underline{x})$ it is necessary to incorporate the constraints imposed on the x_j into the objective function. We implemented this according to the min-max formulation (8). The minimum is $f_3(\underline{-5-\varepsilon})=0$ where $\varepsilon \in [0,0.12]$. The step function exhibits many plateaus which pose a considerable problem for many minimization algorithms.

4) Modified fourth De Jong function (quartic)

$$f_4(\underline{x}) = \sum_{j=0}^{29} (x_j^4 \cdot (j+1) + \eta); \qquad x_j \in [-1.28, 1.28] \qquad (20)$$

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 also included η inside the summation instead of just adding η to the summation result. This change makes $f_4(\underline{x})$ more difficult to minimize. The functional minimum is $f_4(\underline{0}) \leq 30 \cdot E[\eta] = 15$, where $E[\eta]$ is the expectation of η.

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

$$f_5(\underline{x}) = \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{i + \sum_{j=0}^{1} (x_j - a_{ij})^6}} \quad ; \qquad x_j \in [-65.536, 65.536] \tag{21}$$

with a_{i0}={-32, -16,0,16,32} for i = 0,1,2,3,4 and $a_{i0}=a_{i \bmod 5,\, 0}$

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

The global minimum for this function is $f_6(-32,-32) \cong 0.998004$.

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

$$f_6(\underline{x}) = \sum_{j=0}^{3} \begin{cases} 0.15 \cdot (z_j - 0.05 \cdot \mathrm{sgn}(z_j))^2 \cdot d_j & \text{if } \left|x_j - z_j\right| < 0.05 \\ d_j \cdot x_j^2 & \text{otherwise} \end{cases} \quad ; x_j \in [-1000, 1000] \tag{22}$$

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

and d_j = {1,1000,10,100}

$f_6(\underline{x})$ 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 minimum here is $f_6(\underline{x}) = 0$, with $|x_j|<0.05$, j=0,1,2,3.

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

$$f_7(\underline{x}) = \sum_{j=0}^{9} \frac{x_j^2}{4000} - \prod_{j=0}^{9} \cos\left(\frac{x_j}{\sqrt{j+1}}\right) + 1; \qquad x_j \in [-400, 400] \tag{23}$$

Like test function $f_6(\underline{x})$, $f_7(\underline{x})$ has many local minima so that it is very difficult to find the true minimum $f_7(\underline{0}) = 0$.

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

$$f_8(\underline{x}) = 9 - x_0 - x_1; \qquad x_j > 0,\ j=1,2 \tag{24}$$

with $$(x_0 - 3)^2 + (x_1 - 2)^2 \le 16 \tag{25}$$

and $$x_0 \cdot x_1 \le 14 \tag{26}$$

Finding the minimum $f_8(7,2)=0$ poses a special problem, because the minimum is located at the corner of the constrained region defined by (24), (25) and (26). All constraint violations were treated by penalty terms of the form p(δ) = 100 + 100*δ with δ denoting the absolute difference of $f_8(\underline{x})$ to the constraint value. In accordance to (8) the objective function which eventually was evaluated took on the largest value out of $f_8(\underline{x})$ and the penalty terms.

9) Polynomial fitting problem: Storn (1995).

$$f_9(\underline{x}, z) = \sum_{j=0}^{2k} x_j \cdot z^j \text{ , k integer and >0,} \tag{27}$$

is a polynomial of degree 2k in z with the coefficients x_j such that

$$f_9(\underline{x}, z) \in [-1,1] \qquad \text{for} \qquad z \in [-1,1] \tag{28}$$

and $$f_9(\underline{x}, z) \geq T_{2k}(1.2) \qquad \text{for} \qquad z = \pm 1.2 \tag{29}$$

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, $f_9(\underline{x}, z) = T_{2k}(z)$, 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

$$T_8(z) = 1 - 32z^2 + 160z^4 - 256z^6 + 128z^8 \tag{30}$$

with $$T_8(1.2) \cong 72.6606669 \tag{31}$$

as well as

$$T_{16}(z) = 1 - 128z^2 + 2688z^4 - 21504z^6 + 84480z^8 - 180224z^{10} + 212992z^{12} - 131072z^{14} + 32768z^{16} \tag{32}$$

with $$T_{16}(1.2) \cong 10558.1450229 . \tag{33}$$

and used the weighted sum (7) of squared errors in order to transform the above constrained optimization problem into an objective function to be minimized. The weighted sum consisted of 62 samples in case of $T_8(z)$ and 102 samples in case of $T_{16}(z)$. Two samples were placed at z=+/-1.2 respectively, the remaining samples were evenly distributed in the interval $z \in [-1,1]$. The starting values for the parameters were drawn randomly from the interval [-100,100] for (30), (31) and [-1000,1000] for (32), (33).

Test Results

We experimented with each of the four algorithms to find the control settings which provided fastest and smoothest convergence. Table I 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.

$f_i(\underline{x})$	ANM				ASA			DE1				DE2 (F=1)			
i	T	TF	NV	**nfe**	TRS	TAS	**nfe**	NP	F	CR	**nfe**	NP	λ	CR	**nfe**
1	0	n.a.	1	**95**	$1 \cdot 10^{-5}$	10	**397**	10	0.5	0.3	**490**	6	0.95	0.5	**392**
2	0	n.a.	1	**106**	$1 \cdot 10^{-5}$	10000	**11275**	6	0.95	0.5	**746**	6	0.95	0.5	**615**
3	300	0.99	20	**90258**	$1 \cdot 10^{-7}$	100	**354**	10	0.8	0.3	**915**	20	0.95	0.2	**1300**
4	300	0.98	30	**-**	$1 \cdot 10^{-5}$	100	**4812**	10	0.75	0.5	**2378**	10	0.95	0.2	**2873**
5	3000	0.995	50	**-**	$1 \cdot 10^{-5}$	100	**1379**	15	0.9	0.3	**735**	20	0.95	0.2	**828**
6	$5 \cdot 10^{6}$	0.995	100	**-**	$1 \cdot 10^{-5}$	100	**3581**	10	0.4	0.2	**834**	10	0.9	0.2	**1125**
7	10	0.99	50	**-**	$1 \cdot 10^{-5}$	0.1	**-**	30	1.	0.3	**22167**	20	0.99	0.2	**12804**
8	5	0.95	5	**2116**	$1 \cdot 10^{-6}$	300	**11864**	10	0.8	0.5	**1559**	10	0.9	0.9	**1076**
9(k=4)	100	0.95	40	**(391373)**	$1 \cdot 10^{-6}$	1000	**-**	30	0.8	1	**19434**	30	0.6	1.0	**14901**
9(k=8)	$5 \cdot 10^{4}$	0.995	150	**-**	$1 \cdot 10^{-8}$	700	**-**	100	0.65	1	**165680**	80	0.6	1.0	**254824**

Table I: Averaged number of function evaluations (nfe) required for finding the global minimum. A hyphen indicates misconvergence and n.a. stands for "not applicable".

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 twenty test runs with randomly chosen initial parameter vectors for each test function and each minimization.

When the global minimum was 0, we defined the minimization task to be completed once the final value was obtained with an accuracy better than 10^{-6}. For $f_4(\underline{x})$, we chose a value less than 15 to indicate the global minimum and a value less than 0.998004 in the case of $f_5(\underline{x})$.

The results in table I 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. The weighting factors F and λ as well as the crossover constant CR were always chosen from the interval [0,1] with F and λ being >= 0.5 in most cases. According to our experience a reasonable choice for NP is between 3*D and 10*D. These rules of thumb for DE's control variables render DE easy to work with which is one of DE's major assets.

Conclusion and final thoughts

The Differential Evolution method (DE) for minimizing continuous space functions has been introduced and shown to be superior to Adaptive Simulated Annealing (ASA): Ingber (1993), as well as the Annealed Nelder&Mead approach (ANM): Press et alii (1992). DE was the only technique to converge for all of the functions in our test function suite. For those problems where ASA or ANM could find the minimum, DE usually converged faster, especially in the more difficult cases. Since DE is inherently parallel, a further significant speedup can be obtained if the algorithm is executed on a parallel machine or a network of computers. This is especially true for real-world problems where computing the objective function often requires a significant amount of time.

DE requires only few control variables which usually can be drawn from a well-defined numerical interval. This and the fact that DE generates new vectors without resorting to an external probability density function with yet to be chosen mean and standard deviations contributes to the fact that DE is easy to operate. Ease of use is often appreciated in industrial environments, especially in projects where no optimization specialists are present.

Although DE has shown promising results it is still in its infancy and can most probably be improved. 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 travelled through a norrow valley. If the vector population approaches the final minimum, the vector distances decrease, 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 behaviour in real-world applications. The most complex real-world application solved with DE so far is the design of a recursive digital filter with 18 parameters and with multiple constraints and objectives: Storn (1996). Many problems, however, are much larger in scale and DE's behaviour in such cases is still unknown.

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

References

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.

Bunday, B.D. and Garside G.R. (1987), Optimisation Methods in Pascal, Edward Arnold Publishers.

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.

Goldberg, D.E. (1989), Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley.

Griewangk (1981), A.O., Generalized Descent for Global Optimization, JOTA, vol. 34, pp. 11 - 39.

Ingber, L. and Rosen, B. (1992), Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, J. of Mathematical and Computer Modeling, Vol. 16, No. 11, pp. 87 - 100.

Ingber, L. (1993), Simulated Annealing: Practice Versus Theory, J. of Mathematical and Computer Modeling, Vol. 18, No. 11, pp. 29 - 57.

Lueder, E. (1990), Optimization of Circuits with a Large Number of Parameters, Archiv fuer Elektronik und Uebertragungstechnik, Band 44, Heft 2, pp 131 - 138.

Moebus, D. (1990), Algorithmen zur Optimierung von Schaltungen und zur Loesung nichtlinearer Differentialgleichungen, Dissertation am Institut fuer Netzwerk- und Systemtheorie der Univ. Stuttgart.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992), Numerical Recipes in C, Cambridge University Press.

Price, K. (1994), Genetic Annealing, Dr. Dobb's Journal, Oct. 1994, pp. 127 - 132.

Rabiner, L.R. and Gold, B. (1975), Theory and Applications of Digital Signal Processing, Prentice-Hall, Englewood Cliffs, N.J..

Rechenberg, I. (1973), Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart.

Schwefel, H.P. (1995), Evolution and Optimum Seeking, John Wiley.

Storn, R. (1995), Contrained Optimization, Dr. Dobb's Journal, May 1995, pp. 119 - 123.

Storn, R. (1996), Differential Evolution Design of an IIR-Filter, accepted for publication at the IEEE International Conference on Evolutionary Computation (ICEC'96) in Nagoya, Japan, may 1996.

Zimmermann, W. (1990), Operations Research, Oldenbourg.

Figure Captions

Fig.1: Two-dimensional example of an objective function showing its contour lines and the process for generating v in scheme DE1. The weighted difference vector of two arbitrarily chosen vectors is added to a third vector to yield the vector v.

Fig. 2: Illustration of the crossover process for D=7, n=2 and L=3.

Fig.3: Two dimensional example of an objective function showing its contour lines and the process for generating v in scheme DE2.