POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Differential Evolution

K. Ming Leung

Abstract: The differential evolution algorithm for finding the global minimum of a possibly nonlinear and non-differential continuous space function is discussed.

Directory

Last Revision Date: January 21, 2004

Table of Contents

1. Introduction

Price and Storn introduced the Differential Evolution (DE) algorithm several years ago [1, 2, 3] as a new heuristic approach for globally minimizing possibly nonlinear and non-differentiable continuous space functions. Problems involving global minimization over continuous space are ubiquitous throughout the science and engineering communities. DE is a class of floating-point encoded, evolutionary optimization algorithms. It fulfills three very important requirements of any practical optimization technique. DE

1. has the ability to find the global minimum regardless of the initial parameter values
2. can give reasonably fast convergence
3. has only a few controlling parameters so that it is easy to use

Although we will not talk about it here. DE can handle integer and discrete variables, as well as optimization problems where the parameters must obey a set of constraints.

Given a scalar function f that depends on n real parameters, x_j, where $j = 1, 2, \ldots, n$, we are interested in a function of the form:

$$f(\mathbf{x}) : \mathcal{R}^n \rightarrow \mathcal{R}, \text{ where } \mathbf{x} = [x_1, x_2, \ldots, x_n]^T.$$

The goal is to find a set of parameters given by a vector $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ that minimizes the value of this objective (cost) function f. That means that

$$\mathbf{x}^* = \min_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x}).$$

The number of parameters is specified by n. Here we assume that the parameters are real but are not subject to any other constraints.

As with all evolutionary optimization algorithms, DE works with a population of potential solutions, rather than with a single potential solution. Each of the potential solutions in the population is referred to as an individual. Typically the population size, N_p, is held fixed from one generation to the next.

In each generation, the population therefore contains N_p $\mathbf{x}$ vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N_p)}.$$

Each individual vector in the population, $\mathbf{x}^{(i)}$, has n components

$$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T, \qquad i = 1, 2, \ldots, N_p.$$

Each component, $x_j^{(i)}$, represents the j-th parameter (chromosome) of the i-th individual.

The DE algorithm has the following basic steps.

1. Start with an initial population consisting of N_p individuals whose chromosomes are generally chosen randomly.
2. A scheme is used to generate potential off-springs (new $\mathbf{x}$ vectors) from the population, and to select them to go onto the next generation.
3. This process is repeated from one generation to the next until a chosen termination condition is met.

The solution for the minimization problem we sought for is given by the fittest individual (the vector $\mathbf{x}^*$ whose function value $f(\mathbf{x}^*)$ is the lowest) of the final generation.

2. Creating the Initial Population

We start by choosing a population size, N_p. It is a fundamental controlling parameter of the DE algorithm, affecting whether the method will converge or not, and if so, whether it converges to the global minimum. The rate of convergence as well as the efficiency, in terms of the number of function evaluations, are also affected.

It is important to choose N_p to be larger than n, otherwise the population does not have enough genetic diversity and the species can never be able to generate the fittest individual (the optimized solution). Selecting too large a value for N_p decreases the efficiency since it will take more computing time to process each generation.

After choosing N_p, we need to create the initial population. In case we have little idea where the optimized parameters, x_j^* may be, except for the fact that it must lie between a lower bound $x_j^{(L)}$ and an upper bound $x_j^{(U)}$, then we can choose the j-th chromosome randomly in the interval $[x_j^{(L)}, x_j^{(U)}]$. That is, for $i = 1, 2, \ldots, N_p$ and $j = 1, 2, \ldots, n$

we let

$$x_j^{(i)} = x_j^{(L)} + r_j^{(i)} \cdot (x_j^{(U)} - x_j^{(L)})$$

where $r_j^{(i)}$ is a random number distributed statistically randomly and uniformly in the interval $[0, 1]$. Note that when $r_j^{(i)}$ is 0, $x_j^{(i)} = x_j^{(L)}$, and when $r_j^{(i)}$ is 1, $x_j^{(i)} = x_j^{(U)}$. And for $r_j^{(i)}$ somewhere between 0 and 1, $x_j^{(i)}$ is between $x_j^{(L)}$ and $x_j^{(U)}$.

If the actual solution has components that lie outside these lower and upper bounds, it is possible that DE can still converge to the solution, although the rate of convergence may be very slow.

3. Scheme for generating Potential Off-Springs

For each individual in the population, a potential (trial) off-spring for the next generation is created according to the scheme:

$$v_j^{(i)} = \begin{cases} x_j^{(i_1)} + s \cdot (x_j^{(i_2)} - x_j^{(i_3)}) & \text{if } r_j^{(i)} <= c \text{ or } j = n_j^{(i)} \\ x_j^{(i)} & \text{otherwise} \end{cases} \quad (1)$$

Here i_1, i_2 and i_3 are 3 randomly chosen indexes referring to 3 randomly chosen individuals in the current population. They are mutually different from each other and from the running index, i. For each i and j, $r_j^{(i)}$ is a uniformly distributed random real number in $[0, 1]$, and $n_j^{(i)}$ is a randomly chosen integer from 1 to n. The use of $n_j^{(i)}$ is to ensure that at least one chromosome of a potential off-spring differs from its counterpart in the current generation. Parameter s is a real factor in the range $[0, 2]$ that controls the scale of the difference $(x_j^{(i_2)} - x_j^{(i_3)})$ that is added to $x_j^{(i_1)}$ to form a potentially new j-th chromosome for the i-th individual. Parameter c gives the probability that components of a potential off-spring will be accepted. If the component is not accepted, then its value remain unchanged.

When DE converges, a certain fraction of the vectors turn out to be all very close the best vector. It is important to note that the potential off-spring created by these vectors are going to be again almost identical copies of their parents. If this is not true, then the algorithm will unlikely lead to converging results. This is required of any other strategies for creating potential off-springs.

The above equation actually consists of 2 steps. The first step involves generating a new vector with vectors in the current population, and it is referred to as mutation. The second step involves a stochastic process that determine if the component is accepted or rejected. This process is known as crossover.

N_p, s, and c are the 3 fundamental controlling parameter of the DE algorithm. Once their values have been chosen, they remain fixed from one generation to the next. The choice of values for these 3 parameters can affect whether the method will converge or not, and if so, whether it converges to the global minimum. The rate of convergence as well as the number of function evaluations are also affected. Suitable values for these controlling parameters can be found by trial-and-error after a several runs using different values. Fortunately DE is rather robust in that the global minimum can be obtained for s and c varying by $10-20\%$, and even much larger variations in N_p, as long as it is large enough.

DE is so robust that sometimes DE still converges to the correct global minimum even when the computer program contains bugs. Thus one has to exercise extreme care in debugging a DE computer

program.

The above scheme or strategy for creating potential off-spring is known as DE/rand/1/bin. Other strategies exist and have been explored.

4. Selection Scheme for New Off-Springs

The function value (cost) of each potential off-spring is compared with its counterpart in the current population. The one with the lower function value will be included in the population of the next generation. As a result, all the individuals of the next generation are at least as good as their counterparts in the current generation. The interesting point with this selection scheme is that a trial vector is not compared against all the individuals in the current population, but only against its own counterpart in the current population. This selection process is a very greedy one and yet DE can avoid premature convergence to a local minimum and finds the global minimum.

5. Termination Condition

One must be careful in choosing a suitable termination condition in DE. Clearly for each generation one must keep track of the best individual whose function value is the lowest. The solution of the minimization problem is given by the vector of that individual for the final generation. The key point is that best individual often does not change for several generations, even when the best vector is no where close to the ultimate solution. An implementation of the above variant of DE in MATLAB can be found at the website for this course. You can find out the termination condition used there.

6. Other DE Variants or Strategies

Other than the DE variant or strategy we considered above in Eq. (1), many other choices exist for creating potential off-springs. We will mention a few other strategies here.

For each vector in the current population, $\mathbf{x}^{(i)}$, there are a fair number of strategies for generating trial vectors, $\mathbf{u}^{(i)}$. We consider here the following 5 strategies:

$$v_j^{(i)} = x_j^{(\text{best})} + s \cdot (x_j^{(r1)} - x_j^{(r2)}), \tag{2}$$

$$v_j^{(i)} = x_j^{(r_1)} + s \cdot (x_j^{(r2)} - x_j^{(r3)}), \tag{3}$$

$$v_j^{(i)} = x_j^{(i)} + \lambda \cdot (x_j^{(\text{best})} - x_j^{(i)}) + s \cdot (x_j^{(r1)} - x_j^{(r2)}), \tag{4}$$

$$v_j^{(i)} = x_j^{(\text{best})} + s \cdot (x_j^{(r1)} - x_j^{(r2)} + x_j^{(r3)} - x_j^{(r4)}), \tag{5}$$

$$v_j^{(i)} = x_j^{(r_1)} + s \cdot (x_j^{(r2)} - x_j^{(r3)} + x_j^{(r4)} - x_j^{(r5)}), \tag{6}$$

Here $\mathbf{x}^{(r1)}$, $\mathbf{x}^{(r2)}$, $\mathbf{x}^{(r3)}$, $\mathbf{x}^{(r4)}$, and $\mathbf{x}^{(r5)}$ are randomly chosen but mutually distinct vectors in the current population. The current best vector is $\mathbf{x}^{(\text{best})}$. The scales of the difference vectors are controlled by s. For strategy 3, λ controls the basic vector, given by the original vector $\mathbf{x}^{(i)}$ for $\lambda = 0$ and by the best vector $\mathbf{x}^{(\text{best})}$ for $\lambda = 1$.

References

[1] Rainer Storn, "On the usage of differential evolution for function optimization", NAFIPS 1996, Berkeley, pp. 519 - 523, (1996). 3

[2] Rainer Storn and Kenneth Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI, March 1995. (Available via *ftp*). 3

[3] Rainer Storn and Kenneth Price, "Differential Evolution A simple evolution strategy for fast optimization", Dr. Dobb's Journal, April 97, pp. 1824 and p. 78, (1997). 3

[4] Jouni Lampinen and Ivan Zelinka, "Mixed Integer-Discrete-Continuous Optimization By Differential Evolution, Part 1: the optimization method", in Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing, June 9.12. 1999, Brno, Czech Republic, ed. by Pavel Osmera.