

Global Optimization by Differential Evolution

Jouni Lampinen
 Lappeenranta University of Technology
 Department of Information Technology
 Laboratory of Information Processing
 P.O.Box 20, FIN-53851 Lappeenranta, Finland
 phone: +358-5-6243430, fax: +358-5-6243456
 E-mail: jlampine@lut.fi

Abstract

This document briefly introduces the Differential Evolution algorithm for global optimization over continuous spaces.

1. Non-linear global optimization problem

A non-linear global optimization problem can be expressed as follows:

Find

$$X = \{x_1, x_2, x_3, \dots, x_D\}$$

to minimize

$$f(X)$$

subject to constraints

$$g_j(X) \leq 0 \quad j = 1, \dots, m$$

and subject to boundary constraints

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, \dots, D$$

where

$$X \in R$$

(1)

2. Differential Evolution

Price and Storn first introduced the Differential Evolution (DE) algorithm a few years ago [SP95a, SP97b]. DE can be classified as an *evolutionary optimization algorithm*. At present, the best known representatives of this class are *genetic algorithms* [Gol90] and *evolution strategies* [Schw95]. Currently, there are several variants of DE. The particular version to be introduced here is the *DE/rand/1/bin* scheme.

Generally, the function to be optimized, f , is of the form:

$$f(X): R^D \rightarrow R$$

(2)

2(5)

The optimization goal is to minimize the value of this *objective function* $f(X)$,

$$\min(f(X)) \quad (3)$$

by optimizing the values of its parameters:

$$X = (x_1, \dots, x_D), \quad X \in R^D \quad (4)$$

where X denotes a vector composed of D objective function parameters. Usually, the parameters of the objective function are also subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively:

$$x_j^{(L)} \leq x_j \leq x_j^{(U)} \quad j = 1, \dots, D \quad (5)$$

As with all evolutionary optimization algorithms, DE operates on a *population*, P_G , of candidate solutions, not just a single solution. These candidate solutions are the *individuals* of the population. In particular, DE maintains a population of constant size that consists of NP , real-valued vectors, X_{iG} , where i indexes the population and G is the *generation* to which the population belongs.

$$P_G = X_{i,G} \quad i = 1, \dots, NP, \quad G = 1, \dots, G_{max} \quad (6)$$

Additionally, each vector contains D real parameters (*chromosomes* of individuals):

$$X_{i,G} = x_{j,i,G} \quad i = 1, \dots, NP, \quad j = 1, \dots, D \quad (7)$$

In order to establish a starting point for optimum seeking, the population must be initialized. Often there is no more knowledge available about the location of a global optimum than the limits of the problem variables. In this case, a natural way to seed the initial population, $P_{G=0}$, is with random values chosen from within the given boundary constraints:

$$P_0 = x_{j,i,0} = rand_j[0,1] \cdot (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)} \quad i = 1, \dots, NP, \quad j = 1, \dots, D \quad (8)$$

where $rand_j[0,1]$ denotes a uniformly distributed random value within range: $[0.0, 1.0]$ that is chosen anew for each j .

DE's self-referential population reproduction scheme is different from the other evolutionary algorithms (see Figure 1). From the 1st generation forward, vectors in the current population, P_G , are randomly sampled and combined to create candidate vectors for the subsequent generation, P_{G+1} . The population of candidate, or "trial" vectors, $P'_{G+1} = U_{i,G+1} = u_{j,i,G+1}$, is generated as follows:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} = x_{j,r3,G} + F \cdot (x_{j,r1,G} - x_{j,r2,G}) & \text{if } rand_j[0,1] \leq CR \vee j = k \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (9)$$

where

$$i = 1, \dots, NP, \quad j = 1, \dots, D$$

$$k \in \{1, \dots, D\}, \text{ random parameter index, chosen once for each } i$$

$$r_1, r_2, r_3 \in \{1, \dots, NP\}, \text{ randomly selected, except: } r_1 \neq r_2 \neq r_3 \neq i$$

$$CR \in [0,1], \quad F \in (0,1+]$$

The randomly chosen indexes, r_1 , r_2 and r_3 are different from each other and also different from the running index, i . New, random, integer values for r_1 , r_2 and r_3 are chosen for each value of the index i , i.e., for each individual. The index, k , refers to a randomly chosen chromosome which is used to ensure that each individual trial vector, $U_{i,G+1}$, differs from its counterpart in the previous generation, X_{iG} , by at least one parameter. A new, random, integer value is assigned to k prior to the construction of each trial vector, i.e., for each value of the index i .

F and CR are DE control parameters. Like NP , both values remain constant during the search process. F is a real-valued factor in range $(0.0,1.0+]$ that scales the differential variations. The upper limit on F has been empirically determined. So far, it is considered that values of F greater than unity, while possible, do not appear to be productive. However, later on this article it is demonstrated that values higher than $F = 1$, may still be useful.

CR is a real-valued crossover factor in range $[0.0,1.0]$ that controls the probability that a trial vector parameter will come from the randomly chosen, mutated vector, $v_{j,i,G+1}$, instead of from the current vector, $x_{j,i,G}$. Generally, both F and CR affect the convergence velocity and robustness of the search process. Their optimal values are dependent both on objective function characteristics and on the population size, NP . Usually, suitable values for F , CR and NP can be found by trial-and-error after a few tests using different values. Practical advice on how to select control parameters NP , F and CR can be found in [SP95a, Sto96a, SP97a, SP97b], for example. A reasonable first guess is: $F = 0.9$, $CR = 0.9$ and $NP = 10D$.

DE's selection scheme also differs from other evolutionary algorithms. The population for the next generation, P_{G+1} , is selected from the current population, P_G , and the child population, according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} = u_{j,i,G+1} & \text{if } f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (10)$$

Thus, each individual of the temporary population is compared with its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good or better than their counterparts in the current generation. The interesting point concerning DE's selection scheme is that a trial vector is only compared to one individual, not to all the individuals in the current population.

3. Handling simple boundary constraints

In boundary constrained problems, it is essential to ensure that parameter values lie inside their allowed ranges after reproduction. A simple way to guarantee this is to replace parameter values that violate boundary constraints with random values generated within the feasible range:

$$u_{j,i,G+1} = \begin{cases} rand_j[0,1] \cdot (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)} & \text{if } u_{j,i,G+1} < x_j^{(L)} \vee u_{j,i,G+1} > x_j^{(U)} \\ u_{j,i,G+1} & \text{otherwise} \end{cases} \quad (11)$$

where

$$i = 1, \dots, NP, \quad j = 1, \dots, D$$

Another, less efficient method for keeping trial vectors within bounds is to regenerate the offending parameter value according Equation 9 as many times as is necessary to satisfy the boundary constraint. However, in the literature, some other methods have also been proposed.

It should also be mentioned that DE's generating scheme can extend its search beyond the space defined by initial parameter limits (Equations 8 and 9) if allowed to do so. This can be a beneficial property for unconstrained optimization problems because optima that are outside of the initialized range can often be located.

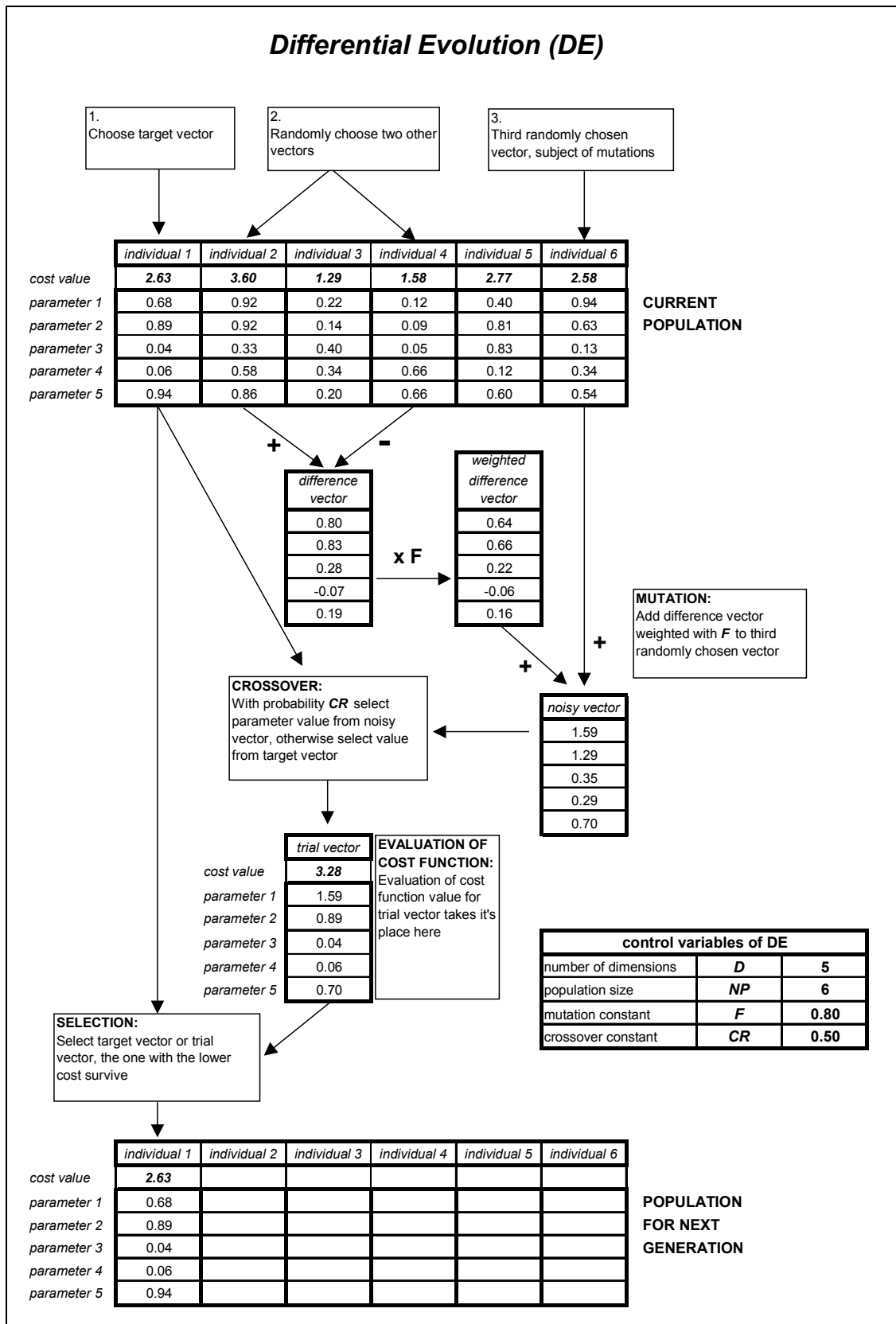


Figure 1. Differential Evolution works directly with the floating-point valued variables of the objective function, not with their (binary) encoding. The functioning of DE is here illustrated in case of a simple objective function $f(X) = x_1 + x_2 + x_3 + x_4 + x_5$ (variables bounded within the range $[0,1]$).

References

- [Gol90] Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading (MA).
- [Lam99a] Lampinen, Jouni (1999). *A Bibliography of Differential Evolution Algorithm*. Technical Report. Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing. Cited 27th December 1999. Available via Internet <http://www.lut.fi/~jlampine/debiblio.htm>.
- [Schw95] Schwefel, Hans-Paul (1995). *Evolution and Optimum Seeking*. John Wiley & Sons Inc., New York.
- [Sto96a] Storn, Rainer (1996). *On the usage of differential evolution for function optimization*. NAFIPS 1996, Berkeley, pp. 519 - 523.
- [SP95a] Storn, Rainer and Price, Kenneth (1995). *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://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z>
- [SP97a] Storn, Rainer and Price, Kenneth (1997). *Differential Evolution – A simple evolution strategy for fast optimization*. Dr. Dobb's Journal, April 97, pp. 18–24 and p. 78.
- [SP97b] Storn, R. and Price, K. (1997). *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*. Journal of Global Optimization, 11(4):341–359, December 1997. Kluwer Academic Publishers.