DIFFERENTIAL EVOLUTION – PARTICLE SWARM OPTIMIZATION

Nuria Gómez Blas, Alberto Arteta, Luis F. de Mingo

***Abstract**: This paper shows the Particle Swarm Optimization algorithm with a Differential Evolution. Each candidate solution is sampled uniformly in [−5,5] ^{D}, whereDdenotes the search space dimension, and the evolution is performed with a classical PSO algorithm and a classical DE/x/1 algorithm according to a random threshold.*

***Keywords**: Benchmarking, Black-box optimization, Direct search, Evolutionary computation, Particle Swarm Optimizacin, Differential Evolution*

***Categories:**G.1.6 [Numerical Analysis]: Optimization-global optimization, unconstrained optimization; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems.*

Introduction

A variety of general search techniques can be employed to locate a solution in a feasible solution space. Most techniques fit into one of the three broad classes. The first major class involves calculus-based techniques. These techniques tend to work quite efficiently on solution spaces with “friendly” landscapes. The second major class involves enumerative techniques, which search (implicitly or explicitly) every point in the solution space. Due to their computational intensity, their usefulness is limited when solving large problems. The third major class of search techniques is the guided random search. Guided searches are similar to enumerative techniques, but they employ heuristics to enhance the search [14].

Evolutionary algorithms (EAs) are one of the most interesting types of guided random search techniques. EAs are a mathematical modeling paradigm inspired by Darwin's theory of evolution. An EA adapts during the search process, using the information it discovers to break the curse of dimensionality that makes non-random and exhaustive search methods computationally intractable. In exchange for their efficiency, most EAs sacrifice the guarantee of locating the global optimum.

Differential evolution (DE) and Particle Swarm Optimization are both stochastic optimization techniques. They produce good results on both real life problems and optimization problems. A simple mixture between those two algorithms, called Differential Evolution – Particle Swarm Optimization (DE-PSO), is explained in the following sections. The explanation will no longer use the sine function, but the more frequently used sphere function. Also note that the explanation for this algorithm will not use a single value, but arrays (vectors) to represent particles and velocities. Therefore it is compatible with more dimensions [2, 3, 4].

Differential evolution

Differential evolution (DE) is a simple evolutionary algorithm for numerical optimization whose most novel feature is that it mutates vectors by adding weighted, random vector differentials to them [1].Differential evolution is, like PSO, a stochastic and population-based optimization technique. It was first introduced in 1996 by Price and Storn

[13]. The algorithm is a modification of genetic annealing, which is beyond the scope of this paper and will not be discussed. Differential evolution is capable of handling non-differentiable, nonlinear and multimodal objective functions and is fairly fast in doing so. DE has participated in the First International IEEE Competition on Evolutionary Optimization (ICEO) and was proven to be one of the fastest evolutionary algorithms [10]. The DE algorithm also works with a population of potential solutions. The principle is the same as PSO: a particle can gain by using information from other particles as well as the results of their own search. However, in the case of differential evolution, that information is sampled randomly. The classical DE algorithm works as shown in the pseudo code of figure 1.

```
1  For each particle
2     Initialize particle and fitness
3  End
4
5  Do
6  For each particle
7     Create mutant
8     Perform crossover
9     If an offspring is better than the parent
10       Replace parent in the next generation
11    Calculate fitness
12 End
13 While maximum iterations is not reached
```

Figure 1: *Pseudo code for DE written in Netbeans.*

Step 1: define a population

For the purpose of this example, a simple 2-dimensional (having 2 variables x1 and x2) sphere function is suitable to search for a minimum. The function rule is as follows:

$$\text{Rand}(0,1) * (\mathbf{b}u - \mathbf{b}l) + \mathbf{b}$$

By drawing the function on a graph it is easily perceived that the global minimum is at coordinates [0,0]. To prove this general truth, a population of 5 particles can be randomly initialized in a search space bounded by [-10,10]. Each particle is presented by an array, or formally speaking by a vector, of D values where D represents the dimension of the search space. Each component of the array can be randomly initialized using equation (a) to initialize a random particle between bounds where rand(0,1) indicates a random number between 0 and 1, bu is

the value of the upper bound and bl is the lower bound. Because the problem requires the search for a global minimum, the particle with the smallest fitness value is closest to the optimum. The table below shows 5 particles and their fitness values upon initialization. The population size does not change during the process.

#	Particle	Particle fitness value
1	[5,-9]	$5^2+(-9)^2=106$
2	[6,1]	$6^2+1^2=37$
3	[-3,5]	$-3^2+5^2=34$
4	[-7,4]	$-7^2+4^2=65$
5	[6,7]	$6^2+7^2=85$

***Table 1**: Each particle is an array of D values where D is the dimension.*

Step 2: mutation

The next step consists of making a mutated vector through differential mutation. It is a process whereby for each particle 3 different randomly chosen particles (r1,r2,r3) create a mutated particle mi (i=1,2,…,population size). This mutated vector is obtained by applying the formula below for each dimension:

$$mij = \mathbf{r}1j + \mathbf{F} * (\mathbf{r}2j - \mathbf{r}3j)$$

With j being the current dimension and F being the positive differential weight value usually between 0 and 1. This value is initialized before the loop and thus the same for every particle in the population. The formula adds the weighted difference between 2 population members to a third member. The table below shows an example of a mutation with the differential weight F set to 1.

Particles for mutation	Mutant vector
2, 4 and 5	[-7,-2]

***Table 2:** Mutant vector for target particle 1 with F=1.*

Step 3: crossover

Crossover is a term used when parameters of the mutant are mixed with parameters of the target vector to form a trial vector z. Which parameters should be crossed is depending on probability.Figure2 shows that 2 out of 5 parameters of the mutant vector were chosen to cross over.

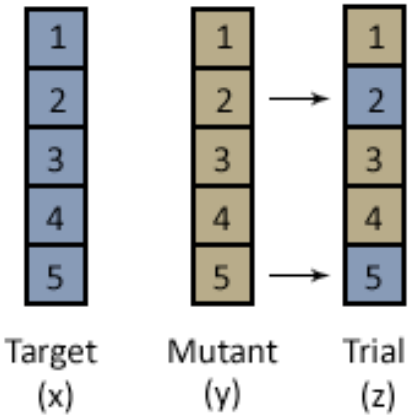

Figure2: Crossover principle.

In Differential Evolution there are 2 main forms of crossover: binomial and exponential. Both ways use a parameter called the Crossover Probability (CR) which is a value between 0 and 1. To understand DE, it is enough to only explain binomial crossover. Many publications explain other ways and should be looked into when more information is necessary.

```
1  For each value i in vector
2    If rand(0, 1) < CR
3      z_i = y_i
4    Else z_i = x_i
5  End
```

Figure3: Pseudo code for binomial crossover (for a minimization)

By applying binomial crossover, one should assign a random value between 0 and 1 and compare it to the crossover probability CR. If CR is bigger, the parameter zi becomes the mutant parameter yi. If it is smaller, zi is equal to the parameter of the current target element, xi, and thus stays the same.

Target particle x	Mutant y for x	Trial vector z
[5,-9]	[-7,-2]	[5,-9]

Table 5: *Binomial crossover for a component of particle 1 with CR=0.5 and twice rand=0.9135 for convenience.*

Step 4: selection

Once the new trial vector z is created, the algorithm has to decide whether or not it should become a member of the new generation (or otherwise known as iteration t+1). This is done by comparing fitness values with the original element of the population. If the fitness of the trial vector is better (i.e. lower for a minimization), itshould be the new member of the population. If it is not better, the current member should be kept unchanged. This process continues, for example, until a maximum number of iterations is exceeded.

Variants of DE

The above mentioned is only the basic variant of DE, but there are many others. In order to classify variants, the following notation is used: DE/x/y/z, with the following components:

X: specifies the vector to be mutated.This can be 'rand' for a randomly chosen population vector or 'best' for the vector with the best fitness value.

Y: The number of difference vectors used.

Z: specifies the crossover scheme. This can be 'bin' for a binomial scheme or 'exp' for an exponential crossover.

Based upon this notation, the version explained in this paper is written as DE/rand/1/bin. DE/best/2/bin is an example of a variant proposed by Price and is proven to be effective. 2 difference vectors are used and seem to improve the diversity of the population if the population size is high enough. For this method, formula (b) can be rewritten:

$$(c)\ vij = \mathbf{x}best + \mathbf{F} * (\mathbf{r}1j + \mathbf{r}2j - \mathbf{r}3j - \mathbf{r}4j)$$

Hybrid DE- PSO

DE-PSO is basically a Differential Evolution algorithm mixed with ideas of Particle Swarm Optimization. It was proposed by Millie Pant, Radha Thangaraj, Crina Grosan and Ajith Abraham in their paper called "Hybrid Differential Evolution – Particle Swarm Optimization Algorithm for Solving Global Optimization Problems". This section explains how the algorithm works according to the very detailed pseudo code presented in figure4, written as a minimizer.

```
For each particle
   Initialize particle and velocity
End
Set Global best and personal best
Do
For each particle
   Choose 3 random particles (r1≠ r2 ≠ r3)
   For each dimension j
      Perform DE part to create mutated U.
   End
   If evaluate (U) < evaluate (X) then
      X = U
   Else
      For each dimension j
         Perform PSO part to create Z
      End
   End
   If evaluate (Z) < evaluate (X) then
      X = Z
   Else
      X = X
   Update personal best
End
Update global best
While stop condition is not reached
```

Figure4: *Pseudo code for DE-PSO*

Step 1: defining population

This step is completed in the same way as PSO and DE. Please note that you should define your population and their velocities as two 2-D arrays when programming. The higher the dimension, the more values each particle will have. A particle in a 2-D search space can be defined as 2 points on a graphs, namely x and y. When you have a particle in a 3-D space, it needs 3 points to be represented on x,y,z axes. To initialize the velocities, one can either choose random values between the predefined bounds, or zero.

Step 2: performing DE

First, 3 random particles (r1,r2 and r3) should be chosen in such a fashion that they are different from each other. The same principle is applied in DE mutation. The second task is to create a mutated value for eachdimension j of the particle. This is done by utilizing equation (b). When the mutation is done for every dimension, one must evaluate the mutated particle with the fitness function, and compare it to the evaluation of the non-mutated,

current particle. If it is smaller, the mutated particle should replace the old one in the population. Though if it is bigger, the particle swarm optimizer should be triggered.

Step 3: performing PSO

If the DE-part of the algorithm did not find a better solution, the PSO is activated. A new particle should be created according to formulas (d) for the velocity and (e) for the new position. A basic velocity and position clamping should be performed here as well. It is enough to just check if the new velocity or position exceeds the bounds in which the algorithm is performed. If the newly created particle is proven to be better, it should be replaced by the old one for the next generation. Both personal best and global best particle vectors should be updated as well. This process is again repeated until the stopping condition is met.

$$(d)\ \boldsymbol{vi}(t+1) = \mathbf{w}^{*}\boldsymbol{vi}\ (t) + \boldsymbol{c1}^{*}\boldsymbol{r1}^{*}(\ \boldsymbol{p}Best - \boldsymbol{xi}) + \mathbf{c}2^{*}\ \boldsymbol{r2}\ ^{*}(\ \boldsymbol{g}Best - \boldsymbol{xi})$$

$$(e)\ \boldsymbol{xi}(t+1) = \boldsymbol{xi}\ (t) + \boldsymbol{vi}\ (t+1)$$

Expansion

More hybrid versions between Particle Swarm Optimization and Differential Evolution have been proposed. One of those is the version proposed by José García, Enrique Alba and Javier Apolloni in their work: " Noiseless Functions Black-Box Optimization: Evaluation of a Hybrid Particle Swarm with Differential Operators". Their model is also simple and is proven to obtain an accurate level of coverage range. The algorithm also contains 2 main parts. The first is the differential variation, where new velocities and positions are calculated according to:

$$(f)\ \mathbf{v'}ij = \mathbf{w} * \mathbf{v}ij + \boldsymbol{\mu} + \boldsymbol{\varphi} * (\mathbf{g}bestj - \mathbf{x}ij)$$

$$(g)\ \mathbf{x'}ij = \mathbf{x}ij + \mathbf{v'}ij$$

Where j is the dimension and i=1,2,…, population size. μ is a scaling factor (μ = UN(0,1)) and φ is the social coefficient (φ = UN(0,1)). The second part is the mutation, which is calculated according to formula (a). It is basically a new particle position between the specified bounds.

Bibliography

[1] S. H. Brooks. A discussion of random methods for seeking maxima.Operations Research, 6:244– 251, 1958.

[2] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009.

[3] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[4] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[5] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives.LargeScaleNonlinearOptimization,pages255–297, 2006.

[6] J. Nelder and R. Mead. The downhill simplex method. Computer Journal, 7:308–313, 1965.

[7] T Jayabarathi, Sandeep Chalasani, Zameer Ahmed Shaik, Nishchal Deep Kodali; "Hybrid Differential Evolution and Particle Swarm Optimization Based Solutions to Short Term Hydro Thermal Scheduling", WSEAS Transactions on Power Systems Issue 11, Volume 2, pp. , ISSN: 1790-5060, 2007.

[8] Piao Haiguo, Wang Zhixin, Zhang Huaqiang, "Cooperative-PSO-Based PID Neural Network Integral Control Strategy and Simulation Research with Asynchronous Motor Controller Design", WSEAS Transactions on Circuits and Systems Volume 8, pp. 136-141, ISSN: 1109-2734, 2009.

[9] Lijia Ren, Xiuchen Jiang, Gehao Sheng, Wu B;"A New Study in Maintenance for Transmission Lines", WSEAS Transactions on Circuits and Systems Volume 7, pp. 53-37, ISSN: 1109-2734, 2008.

[10] Kenneth Price. Differential evolution vs. the functions of the second ICEO. In Proceedings of the IEEE International Congress on Evolutionary Computation, pages 153–157, 1997.

[11] Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). Springer- Verlag New York, Inc., 2005. ISBN 3540209506. URL http://portal.acm.org/citation.cfm?id=1121631.

[12] K.V. Price. Differential evolution: a fast and simple numerical optimizer. In Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American, pages 524–527, 1996. doi: {10.1109/NAFIPS.1996.534790}.

[13] Storn, R., Price, K., "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces", Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, USA (1995).

[14] Paul K. Bergey, and Cliff Ragsdale; Modified differential evolution: a greedy random strategy for genetic recombination. The International Journal of Management Science. Volume 33, Issue 3, June 2005, Pages 255-265. doi:10.1016/j.omega.2004.04.009

Acknowledgment

This work has been partially supported by the Spanish Research Projects:

TRA2010-15645. "COMUNICACIONES EN MALLA PARA VEHICULOS E INFRAESTRUCTURAS INTELIGENTES" (Mesh communication with intelligent vehicles). (2010)

TEC2010-21303-C04-02. "ESTRUCTURAS RESONANTES PARA APLICACIONES DE SEÑAL FOTONICA DE BANDA ANCHA ". (2010).

Authors' Information

Nuria Gómez Blas *– Associate professor U.P.M Crtra Valencia km 7, Madrid-28031, Spain; e-mail: ngomez@eui.upm.es Research: DNA computing, Membrane computing, Education on Applied Mathematics and Informatics*

Albeto Arteta *– Associate professor U.P.M Crtra Valencia km 7, Madrid-28031, Spain; e-mail: aarteta@eui.upm.esResearch: DNA computing, Membrane computing, Education on Applied Mathematics and Informatics*

Luis F. de Mingo *– Associate professor U.P.M Crtra Valencia km 7, Madrid-28031, Spain; e-mail: lfmingo@eui.upm.es Research:,Artificial Intelligence, Social Intelligence, Education on Applied Mathematics and Informatics*