Increasing the Performance of Differential Evolution by Random Number Generation with the Feasibility Region Shape

Felix Calderon, Juan Flores, and Erick De la Vega

Universidad Michoacana de San Nicolás de Hidalgo,
División de Estudios de Posgrado, Facultad de Ingeniería Eléctrica,
Santiago Tapia 403 Centro, Morelia, Michoacán, CP 58000, Mexico
calderon@umich.mx, juanf@umich.mx, edelavega@faraday.fie.umich.mx

Abstract. Global optimization based on evolutionary algorithms can be used for many engineering optimization problems and these algorithms have yielded promising results for solving nonlinear, non-differentiable, and multi-modal optimization problems. In general, evolutionary algorithms require a set of random initial values; in the case of constrained optimization problems, the challenge is to generate random values inside the feasible region. Differential evolution (DE) is a simple and efficient evolutionary algorithm for function optimization over continuous spaces. It outperforms search heuristics when tested over both benchmark and real world problems. DE with Penalty Cost Functions is the most used technique to deal with constraints, but the solution is moved by the penalty factor, losing accuracy. Additionally, the probability to reach the optimal value is near zero for some constrained optimization problems, because the optima of the objective function are located out of the feasible region, therefore the optimal solutions of the problem lie at the border of the feasible region. In this paper we propose an improved DE algorithm for linearly constrained optimization problem. This approach changes the restricted problem to a non-restricted one, since all individuals are generated inside the feasible region. The proposed modification to DE increases the accuracy of the results, compared to DE with penalty Functions; this is accomplished by the generation of random numbers whose convex hull is shaped by the feasible region. We tested our approach with several benchmark and real problems, in particular with problems of economic dispatch of electrical energy.

Keywords: Differential evolution, feasible region, optimization.

1 Introduction

For a differentiable function $f(x) : \mathbb{R}^D \to \mathbb{R}$, computing the minimum $x^* = [x_0^*, x_1^*, \cdots, x_{D-1}^*,]$, can be done using gradient-based methods [1]. Nevertheless, these techniques fail for non-differentiable and discontinuous functions. An alternative is to solve the problem using evolutionary computation; these meta-heuristics compute an initial set of prospect solutions, called initial population.

The most common way to generate a random initial population within a feasible reagion $\mathcal{R}$, is computing a random number with uniform distribution using Equation (1). The population generated this way is uniformly distributed inside a hyper-cube, therefore, when minimizing the function $f(x)$ restricted by constraints of the form $x_j^{min} <= x <= x_j^{max}$ it is very easy to compute the solution using Genetic Algorithms.

$$x_j = x_j^{min} + \alpha * (x_j^{max} - x_j^{min}) \qquad \forall j \in [0, D] \tag{1}$$

Nevertheless, the problem contains a set of linear constraints that limit the region, turning it into a line, a plane, or a hyper-plane (depending on the number of dimensions of the problem). In general, the problem we are to solve using Differential Evolution has the form given by Equation (2)

$$\begin{array}{lll} f(x) \qquad s.a & a_i^T x + b_i >= 0 & \forall i \in [0, M_1 - 1] \\ & a_k^T x + b_k = 0 & \forall k \in [0, M_2 - 1] \end{array} \tag{2}$$

An example of this kind of constrained problem is Economic Dispatch (ED) [2]; ED consists of supplying electrical energy to a load using a set of generation units. Electrical generators have constraints that limit the amount of energy it can provide, and a linear constraint associated with energy conservation. Santos [3] proposes a solution to this problem using Differential Evolution with penalty functions. The problem includes linear constraints and forbidden generation zones.

In this paper, we propose to model the feasible region by a set of points forming the convex hull of the feasible area. Calderon *et al.* [4] presents a GA-based solution using a set of vectors delimiting the feasibility region. Lara *et al* [5] propose an algorithm to compute the convex hull corresponding to an ED problem with a set of inequalities limiting the generation hyper-cube, and one linear constraint – a hyper-plane – where feasible solutions dwell. Based on that set of points, an initial population living inside the feasible region can be computed via linear combinations of points in that convex hull. Nevertheless, Differential Evolution has proven to be more precise than GA [6]. An algorithm for computing a convex hull has exponential time nevertheless given the characteristics for this problem Lara *et al* present an linear algorithm with respect to the intersection points between the hyper plane and the hyper cube (for details see [5]).

To solve a problem like the one pose in Equation (2) using Differential Evolution, we need to guarantee an initial population within the feasible area. Additionally, when the DE operators are applied there is no guarantee for the population to remain within the feasibility region. Our proposal does guarantee both, that the initial population is generated inside and fills uniformly the feasibility region, and that the DE evolution operators keep the population inside that region.

It is very common for evolutionary algorithms to use penalty functions to maintain the population within the feasibility space. The penalty function is charged a cost λ each time a constraint is violated. A problem with penalty functions is that their associated cost moves the solution; besides, if the probability to generate a solution is near zero, the accuracy of the result, in general, is very poor. The basic form of Equation (2) using a penalty function, can be written as in Equation (3).

$$F(x) = f(x) + \lambda \sum_{i=0}^{M-1} C(-a_i^T x - b_i) \tag{3}$$

where $C(y) = 1$ if $y > \tau$ and 0 otherwise; τ is a threshold.

Figure 1(a) shows a rectangle described by the limits $[x_0^{min}, x_0^{max}]$ and $[x_1^{min}, x_1^{max}]$ in two dimensions. That rectangle will contain the initial population if their individuals are generated from Equations (1). Nevertheless, where linear constraints like those in Equation (2) are included in the problem formulation, the feasibility space does not fill that rectangle. Figure 1(b) shows how the area of Figure 1(a) divides in two regions. Let us call $\mathcal{R}_1$ to the unfeasible region and $\mathcal{R}_2$ the feasible region (marked in the figure by numbers 1 and 2, respectively). If we generate a random individual using (1), the probability to generate it within the feasible region is $P(X) = Area(\mathcal{R}_2)/Area(\mathcal{R})$. The probability to generate an individual in the feasible region can decrease for a certain set of constraints; in that case, it is more likely to generate individuals outside the feasible region, or even worse, the probability to generate individuals in the feasible region may be null. A case where the feasible region has zero probability is when we minimize $f(x)$ subject to $x_0^{min} \leq x_0 \leq x_0^{max}$, $x_1^{min} \leq x_1 \leq x_1^{max}$, and $ax_0 + bx_1 = c$.

The rest of the paper is organized as follows. Section 2 describes how to generate a population with the same shape as the feasible region, and proves that the population is indeed inside it. This scheme allows us to change a constrained optimization problem to an unconstrained one. Section 3 presents the details of the DE algorithm and proposes a modification for it to generate all individuals within the feasible region. Section 4 compares solutions obtained by DE with penalty functions with Differential Evolution with Number Generation based on the Feasible Region Shape (DE-NGFRS), as well as the solutions using Mathematica (in some cases), and solutions to problems of economic dispatch. Finally, Section 5 presents the conclusions.

2 Generation of a Random Population with the Shape of the Feasible Region

Number Generation with Feasible Region Shape (NGFRS) is described in this section; NGFRS produces a random population with the shape of the feasible region. We will assume we have a set of vertices $q = \{q_0, q_1, \cdots, q_i, \cdots, q_{N-1}\}$, located on a hyper-plane on D dimensions, which define the convex hull of the feasible region. One way to generate an individual x inside the feasible region is by a linear combination of the vertices, given by Equation (4), as proposed in Calderon *et al.* [4].

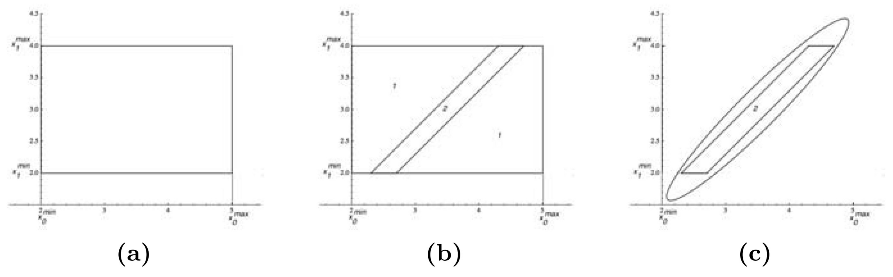

Fig. 1. Feasible Regions generated by a) eq. (1) , b) eq. (2) and c) algorithm 1

$$x = \sum_{i=0}^{N-1} q_i \alpha_i \qquad s.t. \qquad \sum_{i=0}^{N-1} \alpha_i = 1 \tag{4}$$

where α_i is a random weight, such that $\alpha_i \sim U(0, 1)$.

This constraint on the weights generates individuals distributed around the center of the feasible area, reducing the probabilities to generate a random individual at the vertices of the feasible region. This is due to the fact that the sum of random numbers follows a Gaussian distribution, according the central limit theorem [7]. Our proposal consists basically on the generation of random individuals with a normal distribution, covering the feasible area (described by its set of vertices). The mean of the generated population, μ, lies at the center of the feasible region, and the covariance matrix Σ is shaped as an ellipsoid around the feasible region (see Figure 1(c)). The sample mean and covariance of the that distribution is computed from the set of vertices q, using Equations (5), and (6).

$$\mu = \frac{1}{N} \sum_{i=0}^{N-1} q_i \tag{5}$$

$$\Sigma = \frac{1}{N} \sum_{i=0}^{N-1} (q_i - \mu)(q_i - \mu)^T \tag{6}$$

Generating random numbers with zero mean and unity variance such that $z \sim N(0, 1)$ is an easy task. The problem is to generate random numbers with nonzero mean and a covariance different to 1. The expected value of numbers z is $E[z] = 0$ and $E[zz^T] = I$ (I is the identity matrix), given that their mean is zero and their variance is unity. A property of the expected value that will be used to modify the mean and covariance of the randomly generated numbers is given by Equations (7) and (8). To modify the mean of the distribution, we simply add a value μ to numbers z, according to (7). The procedure to modify the covariance is not as straightforward, though.

$$E[z+\mu] = E[z] + \mu = \mu \tag{7}$$
$$E[z\Sigma z^T] = \Sigma E[zz^T] = \Sigma \tag{8}$$

If we represent the covariance matrix in Equation (8) by a singular values decomposition given by $\Sigma = R\Lambda R^T$, where R is a rotation matrix and $\Lambda = diag[\lambda_0, \lambda_1, \cdots, \lambda_{D-1}]$, we can generate numbers that follow a normal distribution $x \sim N(\mu, \Sigma)$, from a random number $z \sim N(0,1)$, those numbers can be generated by making

$$\begin{aligned} E[z\Sigma z^T] = E[zRLL^TR^Tz^T] = E[xx^T] = \Sigma \\ x = RLz + \mu \end{aligned} \tag{9}$$

where $L = diag[\sqrt{\lambda_0}, \sqrt{\lambda_1}, \cdots, \sqrt{\lambda_{D-1}}]$.

This Equation guarantees to produce a random number inside an ellipsoid from the normal distribution. This number is not necessarily within the feasible region, though. To verify that the number is inside the polygon, we take Equations (4) and take them to a matrix form (see Equation 10). The solution by minimum squared can be computed using Equation 11.

$$\begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & \cdots & q_{N-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} \tag{10}$$

$$\begin{aligned} X &= Q\alpha \\ \alpha &= [Q^T * Q]^{-1}Q^TX \end{aligned} \tag{11}$$

So, point x is inside the polygon iff the values of α follow $0 \leq \alpha_i \leq 1$. This procedure is known as computation of the varicentric coordinates [8]. Algorithm 1 summarizes all step described above for Generating Random Numbers inside the Feasible Region. In our test we take advantages of generating the random number with Gaussian distribution instead of uniform distribution because the firsts generate a elliptical shape over the feasible region and the uniform produce a shaped region quite different to the feasible region and in some cases do not cover all the feasible region.

3 Differential Evolution

Differential Evolution (DE) was developed by Storn and Price [9, 6] around 1995 as an efficient and robust meta-heuristic to optimize functions of arbitrary complexity. Like most algorithms in Evolutionary Computation, DE is a population–based optimizer. Most of these methods produce new individuals, by different

Algorithm 1 Algorithm NGFRS

```
Input: $q = \{q_0, q_1, \cdots, q_{N-1}\}$
Output: A set of random numbers $x$

Compute mean $\mu$ and an covariance matrix $\Sigma$ by eq (5) and (6)
Compute eigenvalue decomposition $\Sigma = R \Lambda R^T$
Compute $L = diag[\sqrt{\lambda_0}, \sqrt{\lambda_1}, \cdots, \sqrt{\lambda_{N-1}}]$
For $i = 0, 1, 2, \cdots$
    Generate $z \sim N(0, 1)$
    Compute $x = RLz + \mu$
    Compute the $\alpha$ values solving eq (11)
    If $\alpha_i < 0$ and $\alpha_i > 1$ for one value, reject the number and generate a new value
```

heuristic techniques, as perturbations of old ones (e.g. crossover, mutation, etc.). DE produces new individuals adding the scaled difference of two randomly selected individuals to a third one.

DE maintains two vector populations, containing $N_{pop} \times N_{par}$ real-valued parameters. Population $x^{(g)}$ contains N_{pop} vectors for each generation g, where the kth individual in $x^{(g,k)}$ has N_{par} parameters. Population $v^{(g)}$ are mutant vectors produced from $x^{(g)}$. Each vector in the current population is recombined with a mutant to produce a trial population, $u^{(g)}$. The trial population is stored in the same array as the mutant population, so only two arrays are needed.

The initial population is generated on a searching space $\mathcal{R}$ by random numbers following a uniform probability distribution, with a (possibly) different range for each dimension. Discrete and integral variables are represented by real numbers and then interpreted in the right way. In order to have a Initial population inside the feasible region, instead of the initial population generated by Equation (1), we propose to use Algorithm 1.

Differential mutation adds a scaled randomly chosen, vector difference to a third vector, as you can see in Equation (12)

$$\begin{aligned} v^{(g,k)} &= x^{(g,r_0)} + F(x^{(g,r_1)} - x^{(g,r_2)}) \\ &\forall k \in [1, N_{pop}] \end{aligned} \tag{12}$$

where F is a positive real number that control the rate at which the population evolves. The difference vector indices, r_1 and r_2, and the index r_0 are chosen randomly from $[1, N_{pop}]$.

To complement the differential mutation strategy, DE uses uniform crossover, also known as discrete recombination. Crossover takes place according to Equation (13)

$$\begin{aligned} u_j^{(g,k)} &= \begin{cases} v_j^{(g,k)} & \text{if } rand(0,1) \le Cr \text{ or } j = j_{rand} \\ x_j^{(g,k)} & \text{otherwise} \end{cases} \\ &\forall < j, k > \in [1, N_{par}] \times [1, N_{pop}] \end{aligned} \tag{13}$$

where $Cr \in [0, 1]$ is a user defined parameter that controls the proportion of components copied from the mutant onto the trial vector. $v_j^{(g,k)}$ is the jth component of the ith individual of the gth generation of population $v^{(g)}$.

If the trial vector $u^{(g,k)}$ has an equal or lower fitness value than its target vector $x_j^{(g,k)}$, it replaces the target vector in the next generation (Equation 14).

$$x^{(g+1,k)} = \begin{cases} u^{(g,k)} \text{ if } f(u^{(g,k)}) \leq f(x^{(g,k)}) \\ x^{(g,k)} \text{ otherwise} \end{cases} \quad k \in [1, N_{pop}] \tag{14}$$

Differential Evolution may create a new vector outside of the feasibility region. If so, then reject it and set $x^{(g+1,i)} = x^{(g,i)}$. The complete algorithm for minimizing $f(x)$ inside a feasible region, given by q, is presented in Algorithm 2, which additionally to the DE operation, randomly generates a new random number if a probability $\gamma \leq 0.5$. With this condition it is more likely for the population to reach the global minimum inside the feasible region.

Algorithm 2 DE-NGFRS

```
Input: $q$, $N_{pop}$, $N_{gen}$, $F$, $C_r$ and $f(x)$
Output: $x^{(*,*)}$

Compute a initial population $x^{(g)}$ by algorithm 1, with mean $\overline{q}$ and compute
the fitness fuction $f(x)$ with out restriction, for each population member
For $g = 0, 1, 2, \cdots, N_{gen}$ {
    For $k = 0, 1, 2, \cdots, N_{pop}$ {
        Generate $\gamma \sim U(0, 1)$
        if $\gamma < 0.5$ {
            Compute $v^{(g,k)}$ by Equation (12)
            Compute the crossover $u^{(g,k)}$ by Equation (13)
            Select $x^{(g+1,k)}$ by (14)
            If $x^{(g+1,k)}$ is out of the feasible region set $x^{(g+1,k)} = x^{(g,k)}$
        }
        else compute $x^{(g+1,k)}$ by algorithm 1 with mean $x^{(g,k)}$
    }
}
```

4 Results

To show our algorithm's performance, we test it with five functions. The first one is a sixth degree two-dimensional function; the second one is the Rosenbrock function, and the remaining three functions are the objective functions belonging to the Economic Dispatch problems known as IEEE14 [10], Wong [12] and Wood [11]. The following subsections present the details of these experiments.

4.1 Sixth Degree Function

This first experiment aims to solve the function $f_1(x)$ (15). The convex hull of the feasible region was determined computing the intersection of the search space with the linear constraint of the optimization problem. The set of vertices of the convex hull is $q = \{[-23, -95], [-24, -94], [-20, -50], [19, 1], [20, -2], [10, -40]\}$. To solve this problem using DE with a penalty function (3) the cost associated to violating a constraint was $\lambda = 1 \times 10^{10}$. Table 1 presents the best, mean, and worst case of 100 independent runs of Algorithm DE-NGFRS. The same problem was presented to Mathematica, which failed to compute a solution. DE with a penalty function converges in some cases and it does not in others.

$$\begin{aligned} f_1(x) = -3x^6 + 2x^5 - x + 2y^3 + 45y + 23 \qquad (15)\\ s.a\\ x + y + 118 \geq 0, 11x - y + 170 \geq 0, 17x - 13y - 310 \geq 0\\ -3x - y + 58 \geq 0, -19x + 5y + 39 \geq 0, -5x + 170 + 3y \geq 0 \end{aligned}$$

Table 1. Sixth Degree Function results for one hundred independent runs

Algorithm	**values**	**time (sec)**	$\boldsymbol{x}$	$\boldsymbol{f_1(x)}$
DE-NGFRS	Min	0.0203	[-24.0000,-94.0000]	-5.90900×10^8
	Mean	0.0203	[-24.0000,-94.0000]	-5.90900×10^8
	Max	0.0203	[-23.9983,-93.9818]	-5.90900×10^8
Mathematica	Always	10.73650	[1.5250, 0.0000]	0.234956
Pelnalty Function	Min	0.046	[-24.0000,-94.0000]	-5.9090×10^8
	Mean	0.049	$[-1.58 \times 10^{22}, -1.12 \times 10^{47}]$	-2.8228×10^{145}
	Max	0.048,	$[-1.48 \times 10^{24}, -1.11 \times 10^{49}]$	-2.82282×10^{147}

4.2 Rosenbrock Function

This experiment solves a problem using the Rosenbrock function constrained to a thin band described by Equation $f_2(x)$ (16). The vertices of the convex hull of the feasible region are $q = \{[2, 4], [-39, 101], [-40, 100], [1, 3]\}$ and the cost for the penalty function was $\lambda = 100$ (3). Table 2 shows the results; our proposal outperforms the other schemes in every case. DE with a penalty function converges only in some cases, and Mathematica presents similar results with executions times greater than those taken by Algorithm DE-NGFRS.

$$\begin{aligned} f_2(x) = (1 - x_0)^2 + 100(x_0 - x_1^2)^2 \qquad (16)\\ s.a \end{aligned}$$

$$-97.0x_0 - 41.0x_1 + 358 \geq 0, x_0 - x_1 + 140.0 \geq 0$$
$$97.0x_0 + 41.0x_1 - 220 \geq 0, -x_0 + x_1 - 2.0 \geq 0$$

Table 2. Results for Rosenbrock Funcion, for one hundred independent runs

Algorithm	**values**	**time (sec)**	x	$f_2(x)$
DE-NGFRS	min	0.484	[1.99889, 3.99889]	0.998889
	mean	0.484	[1.99889, 3.99889]	0.998889
	max	0.484	[1.99880, 3.99889]	0.998889
Mathematica	Always	8.07094	[1.99888, 3.99888]	0.998889
Penalty	min	2.089	[1.99889, 3.99889]	0.998889
	mean	2.151	[-3.76033, 17.30290]	48.9379
	Max	2.156	[-7.58911, 57.60030]	173.776

4.3 Economic Dispatch

This subsection presents the results obtained by solving the economic dispatch problem with three electrical networks corresponding to IEEE14 (see [10]), Wood (see [11]), and Wong (see [12]). The cost (objective) functions for each of the networks are given by Equations (17), (18), and (19), respectively. The sets of vertices of the convex hulls of the corresponding feasible regions were computed using the algorithm proposed by Lara *et al.* [5]; those vectores are shown on Table 3. The set of convex hull vectors is computed and then we proceed to solve the unconstrained version of the optimization problem using Algorihm 2. The average time to compute those vectors was 20 ms.

Table 4 shows the comparative results for these networks. The table shows that we solve the economic dispatch in a time much lower than those reported in [4] producing results with better accuracy. See the results for the Wong problem using Algorithm DE-NGFRS.

$$\begin{aligned} f_{IEEE14}(x) = {} & 2\times10^{-5} + 0.003x_0 + 0.01x_0^2 + 2\times10^{-5} + 0.003x_1 + 0.01x_1^2 \\ & +2\times10^{-5} + 0.003x_2 + 0.01x_2^2 + 2\times10^{-5} + 0.003x_3 + 0.01x_3^2 \\ & +2\times10^{-5} + 0.003x_4 + 0.01x_4^2 \\ & s.a. \\ & 10 \leq x_0 \leq 80, \quad 10 \leq x_1 \leq 60, \quad 10 \leq x_2 \leq 60, \quad 10 \leq x_3 \leq 60, \\ & 10 \leq x_4 \leq 80, \quad and \quad x_0 + x_1 + x_2 + x_3 + x_4 = 300.5576 \end{aligned} \tag{17}$$

$$f_{Wood}(x) = 749.55 + 6.950x_0 + 9.680\times10^{-4}x_0^2 + 1.270\times10^{-7}x_0^3 \tag{18}$$

$$\begin{aligned} &+1285.00 + 7.051x_1 + 7.375 \times 10^{-4}x_1^2 + 6.453 \times 10^{-8}x_1^3 \\ &+1531.00 + 6.531x_2 + 1.040 \times 10^{-3}x_2^2 + 9.980 \times 10^{-8}x_2^3 \\ &s.a \\ &320 \leq x_0 \leq 800, \quad 300 \leq x_1 \leq 1200, \quad 275 \leq x_2 \leq 1100 \\ &and \quad x_0 + x_1 + x_2 = 2500 \end{aligned}$$

$$\begin{aligned} f_{Wong}(x) = 11.20 &+ 5.10238x_0 - 2.64290 \times 10^{-3}x_0^2 + 3.3333 \times 10^{-6}x_0^3 \\ &-632.00 + 13.01x_1 - 3.05714 \times 10^{-2}x_1^2 + 3.3333 \times 10^{-5}x_1^3 \\ &+147.144 + 4.28997x_2 + 3.08450 \times 10^{-}4x_2^2 - 1.7677 \times 10^{-7}x_2^3 \\ &s.a \\ &100 \leq x_0 \leq 500, \quad 100 \leq x_1 \leq 500, \quad 200 \leq x_2 \leq 1000 \\ &and \quad x_0 + x_1 + x_2 = 1443.4 \end{aligned} \tag{19}$$

Table 5 shows the parameters used to perform these tests, the average and standard deviation of the error function of 100 independent runs for the five test problems. Table 5 includes the name, population size N_{pop}, the number of generations N_{gen}, the mutation parameter F, the crossover parameter C_r, and the mean $\overline{f(x)}$ and standard deviation $EstDev(f(x))$ of the error function $f(x)$. Note that the mean of the error function is consistent in all experiments, with a very low standard deviation. The population size and generation number depend of the number of variables and this parameter is hand picked for the best results in general we can use the biggest for this parameters with the same results, obviously the execution time will be quite different.

5 Conclusions

Evolutionary computation has proven to be a good tool to solve non-linear optimization problems, exhibiting an advantage over traditional gradient-based methods, specially for discontinuous and non-differentiable objective functions. These difficult problems become even harder when constraints are added to the problem. Those constraints often represent relations that have to be preserved among the problem variables, energy or flow conservation, etc. One way of dealing with constrained optimization problems is to add penalties to the objective function when an individual lies outside the feasible region. Unfortunately, this approach leads to time wasted in generating and discarding individuals outside the feasible regions. There are situations where the proportion of the size of the feasible region, with respect to the search space is zero. This fact affects an evolutionary search from the moment of generating the initial population, and later on the individuals produced by evolutionary operations; in those cases most time is wasted in discarding individuals violating the constraints, and the population gets to an impasse.

Table 3. Convex Hull Vectors computed by Lara *et al.* [5]

Network	**Active Power Vectors in KW**
IEEE 14	$v_1 = [40.5576, 60.0000, 60.0000, 60.0000, 80.0000]^T$
	$v_2 = [80.0000, 20.5576, 60.0000, 60.0000, 80.0000]^T$
	$v_3 = [80.0000, 60.0000, 20.5576, 60.0000, 80.0000]^T$
	$v_4 = [80.0000, 60.0000, 60.0000, 20.5576, 80.0000]^T$
	$v_5 = [80.0000, 60.0000, 60.0000, 60.0000, 40.5576]^T$
Wood	$v_1 = [320.0, 1080.0, 1100.0]^T$
	$v_2 = [800.0, 600.0, 1100.0]^T$
	$v_3 = [320.0, 1200.0, 980.0]^T$
	$v_4 = [800.0, 1200.0, 500.0]^T$
Wong	$v_1 = [343.40, 100.00, 1000.00]^T$
	$v_2 = [100.00, 343.40, 1000.00]^T$
	$v_3 = [100.00, 500.00, 843.40]^T$
	$v_4 = [500.00, 500.00, 443.40]^T$
	$v_5 = [500.00, 100.00, 843.40]^T$

Table 4. Comparative Results for Economic Dispatch problems.

Network	**Algorithm**	**Demand**	**Generation Cost**	**Time (seg.)**	**Active Power Vector Solution**
IEEE 14	[10]	300.5576	181.5724	-	[60.2788 60.0000 ... 60.0000 60.0000 60.2788]
	[4]	300.5576	181.5724	4.276	[60.2789 59.9999 ... 59.9999 59.9997 60.2789]
	DE-NGFRS	300.5576	181.5724	0.811	[60.2788 60.0000 ... 60.0000 60.0000 60.2788
Wood	[11]	2500.1000	22730.21669	-	[726.9000 912.8000 860.4000]
	[4]	2500.0000	22729.32458	2.814	[725.0078 910.1251 864.8670]
	DE-NGFRS	2500.0000	22729.32458	0.027	[724.9915 910.1534 864.8551]
Wong	[12]	1462.4480	6639.50400	-	[376.1226 100.0521 986.2728]
	[4]	1443.4000	6552.23790	2.842	[343.3980 100.0415 999.9604]
	DE-NGFRS	1443.4000	6552.09315	0.260	[343.4000 100.0000 1000.0000]

Table 5. Parameters, Average and Standard Deviation for all the examples

Example	N_{pop}	N_{gen}	F	C_r	$\overline{f(x)}$	$EstDev(f(x))$
G6	50	500	0.95	0.90	-5.9090×10^8	0.01118
Rosenbrock	1500	500	1.2	0.90	0.9988	1.8058×10^{-15}
IEEE14	1000	1000	0.95	0.90	181.5724	3.8849×10^{-12}
Wood	50	700	0.95	0.90	22729.3246	2.9451×10^{-11}
Wong	500	700	0.95	0.90	6552.0931	1.6024×10^{-11}

In this paper we present an algorithm called NGFRS, which allows us to generate a population of individuals that fills a convex area delimited by linear constraints. The same ideas of NGFRS have been applied to solve non-linear, linearly constrained optimization problems; the resulting algorithm is called DE-NGFRS. It has been empirically shown that NGFRS exhibits a performance way superior to DE with a penalty function. The results obtained in the solution of economic dispatch problems using DE-NGFRS was compared with the results presented for those problems in previous work; DE-NGFRS proved to improve those results. Additionally, DE-NGFRS proves to be better than Algorithm GA-V, which is based on Genetic Algorithms.

References

1. Jorge, N., Wright, S.J.: Numerical Optimization. Springer (2000)
2. Stevenson, W.D.: Elements of Power System Analysis. Mc Graw Hill (1982)
3. dos Santos Coelhoa, L., Marianib, V.C.: Improved differential evolution algorithms for handling economic dispatch optimization with generator constraints. Energy Conversion and Management, **48**, 1631–1639 (2007)
4. Calderon Felix, Fuerte-Esquivel Claudio R, S.J., J., F.J.: A constraint-handling genetic algorithm to power economic dispatch. MICAI 2008: Advances in Artificial Intelligence Lecture Notes in Computer Science, 371–381 (2008)
5. Lara, C., Flores, J.J., Calderon, F.: On the hyperbox hyperplane intersection problem. Infocomp Computer Journal Computer Science, **8**, 21–27 (2009)
6. Storn, R., Price, K., Lampinen, J.: Differential Evolution. A Practical Approach to Global Optimization. Springer–Verlag, Berlin, Germany (2005)
7. Fischer, H.: A History of the Central Limit Theorem: From Classical to Modern Probability Theory. Springer (2010)
8. Lawson, C.L.: Properties of n-dimensional triangulations. Computer-Aided Geometric Design, **3**, 231–246 (1986)
9. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report 95-012, ICSI (1995)
10. Ongsakul, W., Ruangpayoongsak, N.: Constrained dynamic economic dispatch by simulatedannealing/genetic algorithms. In: Power Industry Computer Applications, 2001. PICA 2001. Innovative Computing for Power - Electric Energy Meets the Market. 22nd IEEE Power Engineering Society International Conference, 207–212 (2001)
11. Wood, A., Wollenberg, B.: Power Generation, Operation, and Control. John Wiley and Sons Inc (1984)
12. Wong, K., Fung, C.: Simulated annealing based economic dispatch algorithm. In: Generation, Transmission and Distribution, IEE Proceedings C. Volume 140, 509–515 (1993)