EA Techniques for Optimal Power Flow. Parameter Tuning by Mathematical Test Functions

Florin Solomonesc, Constantin Barbulescu, Stefan Kilyeni, Oana Pop, Marius Cornoiu, Adrian Olariu

"Politehnica" University of Timisoara, 2 Bd. V. Parvan, Timisoara, Romania, claudiu.solomonesc@upt.ro, constantin.barbulescu@upt.ro, stefan.kilyeni@upt.ro, oana.pop@upt.ro, marius.cornoiu@upt.ro, adrian.olariu@upt.ro

Abstract: The goal of this paper is to establish the best values for evolutionary algorithm main parameters. To do this, a real coded algorithm is tested on Rosenbrock, Schwefel and Rastrigin functions. Different genetic operators' implementation ways are described. Several numbers of variables have been analyzed. An original software tool has been developed in a Matlab environment. The most complex three mathematical test functions have been implemented within the software tool; these functions are considered as ideal for studying the behaviour of the algorithm. The settings established are crucial in the optimal power flow computing and the transmission network expansion planning, by means of EA techniques.

Keywords: evolutionary algorithm; test function; crossover rate; mutation rate; random mutation; variable step mutation

1 Introduction

A great amount of attention is paid to Evolutionary Algorithms (EAs) for engineering problem-solving. These algorithms are defined in [1] as population-based meta-heuristic optimization algorithms, having genetics-inspired mechanisms to iteratively refine a set of solution candidates. While the solutions obtained are evaluated according to traditional methods, EAs eliminate complex mathematical computations. Evolutionary Algorithms include the following categories: genetic algorithms (GAs), evolutionary programming (EP) and evolutionary strategies (ESs). The GAs usually use binary digit arrays, whereas the ESs, decimal variable arrays. However, the GAs variables are coded as decimal numbers in several papers.

The main concepts that have been adapted from genetics can be described as follows:

Concept	**Meaning in genetics**	**Mathematical optimization**
Phenotype	Set of visible features of an individual; these features are developed on hereditary basis under environmental constraints;	A possible solution;

Concept	**Meaning in genetics**	**Mathematical optimization**
Genotype	Set of hereditary proprieties of an organism. Also referred to as a *chromosome*;	A set of variables, put in a form that can be easy processed by the optimization algorithm. usually represented as an array;
Phenome	A group of all phenotypes that define an organ or an organism;	Solution space;
Genome	The group of all different chromosomes;	Search space;
Gene	The basic unit of a chromosome, which carries the hereditary properties;	A decimal or binary variable;
Locus	The position of a gene in a chromosome;	The variable position within an array;
Alela	One of the forms a gene can take;	The variable value;
Population	A set of chromosomes;	A small number of variable sets generated or selected from the search space;
Generation	A population with members of same age.	The population on which the genetic operators are applied during iteration.

Special interest in this kind of optimization methods has been shown in the field of electrical power engineering. A binary coded genetic algorithm for optimal power flow (OPF) is described in [2], based on the FACTS device control (Flexible Alternative Current Transmission Systems). Good results have been obtained for the IEEE14 test power system. [3] describes a binary GA that uses both continuous (real power and bus voltage) and discrete variables (transformer tap ratio) to solve the OPF. The algorithm is tested on IEEE30 and IEEE_3Area_96 systems. [4] proposes a new GA initialization procedure used to determine the OPF. This procedure relies on a voltage grading technique. The OPF problem is also discussed in [5], taking in to account multiple contingencies. A cost efficient OPF method is presented in [6], where the EA uses an adaptive mutation rate. The results refer to the IEEE30 test system, as well as to two real power systems. In [7], the Matlab-integrated genetic algorithm toolbox is used to solve the OPF. The results for the IEEE30 test system are compared to the ones arrived at by means of a particle swarm optimization (PSO) algorithm.

Kazemi et al. [8] put forward a method based on a decimal evolutionary algorithm, to solve the transmission network expansion planning (TNEP). The algorithm has been tested on a modified 6-bus Garver system. In [9], a GA is employed to determine the optimal expansion plan taking into consideration the power loss minimization and the total investment cost for the Azerbaijani power grid. In [10], a method for static TNEP, under deregulation, is presented. The method is based on a GA and a fuzzy decision-making procedure. An improved version of this method, which can deal with dynamic TNEP, is provided in [11]. For the short-term TNEP, a GA combined with a hill-climbing technique is discussed in [12]. Hill-climbing is used for mutation and leads to faster convergence. In [13] and [14], the Pareto dominance is applied to compare the solutions of a multi-objective GA for TNEP.

In [15], the transmission expansion is solved in relation to load uncertainty, with the help of scenario-based GA.

The generation expansion planning (GEP) is also approached using EAs. In [16], the expansion of thermal power plants is discussed taking into account the emissions and their long-term effects. The algorithm is a combination of GA and Bender's decomposition method. in [17] and [18], Murugan et al. apply an NSGA-II (non-dominated sorting genetic algorithm) to solve the GEP for the IEEE30 system. The distribution network expansion is approached by Wang in [19], [20].

The goal of this paper is twofold: to analyze the behaviour of EAs and to determine the optimal values of the main parameters (e.g. population size, mutation and crossover ratios, etc.). The results will be employed in OPF computing and in TNEP, by means of EA techniques. The real coded algorithm is tested on three multiple variable functions: Schwefel, Rastrigin and Rosenbrock. Section 2 describes the implementation of the algorithm. The results are then discussed in section 3.

2 Evolutionary Algorithm (EA) for Test Functions

The flow chart of the proposed EA is presented in Fig. 1. Basically, the algorithm starts from a randomly generated initial population, which becomes the current population for the first iteration (first generation). The genetic operators are applied within each iteration for the current population. The computation process finishes when at least one of the stopping criteria has been fulfilled. In what follows, every step presented in Fig. 1 is described in detail.

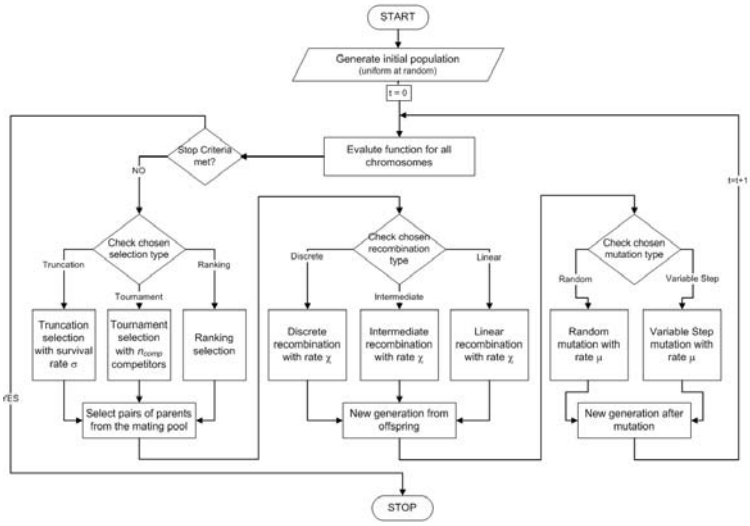


Figure 1
Evolutionary algorithm flow chart

2.1 Variable encoding

The variables are stored within a structure called *chromosome*, which is represented as an array. Each variable, whether discrete or continuous, occupies a number of elements from the array (these elements are called *genes*). If binary representation is used, then the number of genes for each variable is set according to the function domain and the desired precision. In this paper, real representation is used, so every gene represents a variable. The chromosome can be represented as the set $\boldsymbol{x} = \{x_1, x_2, ..., x_d\}$, where d stands for the number of variables. Fig. 2 shows an example of such a chromosome, used to optimize the Schwefel function with six variables. According to [21], this function can take any number of variables and is defined within the [-500; 500] interval.

-61,2556	251,2671	-148,34	-337,818	-393,347	353,0311

Figure 2
Chromosome structure example

2.2 Population size and initial population

Several chromosomes (individuals) form the population. A t moment population is called a *generation*, where t stands for the generation counter. The population size n_c, which is considered as fixed during one run of the algorithm, is one of the parameters that are studied within this paper. The initial population $\boldsymbol{x}^0$ is determined using Eq. (1).

$$\begin{aligned} &\boldsymbol{x}^0_{i,j} = (x_{max} - x_{min}) \cdot a_{i,j} + x_{min} \qquad i = 1, 2, ..., n_c \\ &a_{i,j} \in [0;1], \text{ uniform at } random \qquad j = 1, 2, ..., d \end{aligned} \tag{1}$$

As it can be seen from Eq. (1), the initial population has a uniform random distribution between the domain limits [x_{min}; x_{max}] and is generated without other constraints. It may be necessary to consider several constraints when generating the initial population, so that the algorithm starts from a set of feasible solutions.

2.3 Evaluation and selection

The function value is evaluated for each chromosome within the population. Thus, for each $\boldsymbol{x}_i$ array, there will be a $f_i = f(\boldsymbol{x}_i)$ value. In engineering, most of the optimization problems are minimization problems, the smallest f_i value being the best.

Selection ensures that the best fitted individuals have a chance to produce offsprings for the next generation, but it also offers a small chance to the others to be selected as well. The individuals are first selected according to the function values and then copied in a buffer matrix called *the mating pool* (***MP***). Three selection types have been implemented: tournament, truncation and ranking.

Truncation selection is very easy to apply. The corresponding parameter is the *survival rate* σ and it represents the population percentage that will be selected for

reproduction. This parameter is conventionally inputted as a number between 0 and 1 (e.g. 0.5 represents half of the population). If the survival rate is low (e.g. 0.25), there is a risk of losing diversity and if it is high (e.g. 0.75), convergence may be slow.

The stages of this selection are the following:

- the population is *sorted ascending* according to the function values;
- the best $n_{sel} = \sigma \cdot n_c$ chromosomes are copied in the mating pool;
- copies of these chromosomes are made until the mating pool is full (the number of chromosomes in the mating pool equals n_c).

Given that the truncation selection is an iterative process, the following relations may be established for iteration k:

$$\begin{aligned} & f(\boldsymbol{x}_{sel}^k) = \min\{f(\boldsymbol{x}_i^k)\} \qquad i = 1,2,...,n_c + 1 - k, \\ & \boldsymbol{MP}^k = \boldsymbol{MP}^{k-1} \cup \boldsymbol{x}_{sel}^k \qquad k = 1,2,...,n_{sel} \\ & \boldsymbol{P}^{k+1} = \boldsymbol{P}^k \setminus \boldsymbol{x}_{sel}^k \end{aligned} \tag{2}$$

Tournament selection finds the fittest individual in a small group randomly selected from the current population. The parameter n_{comp} designates the number of competitors (the size of the group of chromosomes to be compared).

This is an iterative process, which comprises the following stages:

- a group of n_{comp} chromosomes is randomly chosen form the current population ($\boldsymbol{x}_{comp_i}$, i = 1,2,...,n_{comp});
- the function values for each chromosome are compared to find their minimum;
- the chromosome corresponding to the minimum function value is copied in the mating pool;
- the process stops if the size of the mating pool equals n_c.

The following relations may be written for iteration k:

$$\begin{aligned} & a_i^k \in \{1,2,...,n_c\}, \text{ at random} \\ & \boldsymbol{x}_{comp_i}^k = \boldsymbol{x}_{a_i}^k \qquad i = 1,2,...,n_{comp} \\ & f(\boldsymbol{x}_{sel}^k) = \min\{f(\boldsymbol{x}_{comp_i}^k)\} \qquad k = 1,2,...,n_c \\ & \boldsymbol{MP}^k = \boldsymbol{MP}^{k-1} \cup \boldsymbol{x}_{sel}^k \\ & \boldsymbol{P}^{k+1} = \boldsymbol{P}^k \end{aligned} \tag{3}$$

Note that a chromosome can be selected several times. If parameter n_{comp} has a high value, it may lose diversity, because a good chromosome is chosen too many times. However, if it is too low, suitable chromosomes may be omitted in the selection process.

Ranking selection is derived from roulette wheel selection. Unlike the latter, it can manage negative function values, by working with *ranks* instead. For a better understanding, a detailed description of each stage of this process is presented below:

- the population is *sorted descending* according to the function values;
- each chromosome in the sorted population is ranked from *1* to n_c - *1* being the worst and being n_c best:

$$rank_i = rank(\boldsymbol{x}_i) = i \qquad i = 1, 2, ..., n_C \tag{4}$$

- the selection probability of a chromosome is computed:

$$p_i = p(\boldsymbol{x}_i) = rank_i / \sum_i rank_i \qquad i = 1, 2, ..., n_C \tag{5}$$

- the cumulative sum of these probabilities is expressed as:

$$Cs_i = Cs(\boldsymbol{x}_i) = \sum_{j=1}^{i} p_j \qquad i = 1, 2, ..., n_C \tag{6}$$

- a number is randomly generated within the interval [*0*; *1*); the first chromosome that has the corresponding cumulative sum of probabilities greater than the random number is copied in the mating pool;
- the process stops if the size of the mating pool equals n_c.

As in the previous case, a chromosome can be selected several times. This is an iterative process. The following relations may be written for iteration *k*:

$$\begin{aligned} & a \in [0;1], \text{ uniform and random} \\ & Cs(\boldsymbol{x}_{sel}^k) = \min\{Cs(\boldsymbol{x}_i^k) \mid Cs(\boldsymbol{x}_i^k) > a\} \qquad i = 1, 2, ..., n_C \\ & \boldsymbol{MP}^k = \boldsymbol{MP}^{k-1} \cup \boldsymbol{x}_{sel}^k \qquad k = 1, 2, ..., n_C \\ & \boldsymbol{P}^{k+1} = \boldsymbol{P}^k \end{aligned} \tag{7}$$

2.4 Mating and Crossover

The next step in the algorithm is to create pairs of parents (mating) from the chromosomes copied in the mating pool. The pairs are chosen by randomly selecting two different positions in the mating pool until $n_C / 2$ pairs are created. Various situations may arise: a chromosome is never chosen to mate; a chromosome is chosen to mate several times; if a chromosome was selected several times, it may mate with another copy of itself. Not all the pairs of parents will undergo crossing-over; some will be copied in the new generation as they are. The number of pairs that will undergo crossing-over is given by the *crossover rate* χ, which is conventionally considered as a number between 0 and 1. In this paper, the crossover rate ranges from 0.5 to 0.9.

Crossing-over generates two new sets of variables from two sets of parent variables, based on a mathematical relation. Three crossover types are discussed in this

paper: discrete (uniform), intermediate and linear. The number of pairs that undergo crossing-over is $n_{rp} = \chi \cdot n_c / 2$, irrespective of the type used.

All the three crossover types are based on relations 8 [22]

$$\begin{aligned} \boldsymbol{x}^{o1} &= \left\{ r \cdot x_i^M + (1-r) \cdot x_i^F \right\} \\ \boldsymbol{x}^{o2} &= \left\{ r \cdot x_i^F + (1-r) \cdot x_i^M \right\} \end{aligned} \qquad i = 1, 2, ..., d \tag{8}$$

where $\boldsymbol{x}^M, \boldsymbol{x}^F$ – parents and $\boldsymbol{x}^{o1}, \boldsymbol{x}^{o2}$ – offsprings.

Variable *r* is randomly generated. The way it is represented as well as the sets and intervals of its values defines the crossover type. For *discrete crossover*, *r* is a *2 x d* matrix containing zeros and ones, as illustrated by the following relation:

$$\boldsymbol{r} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,d} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,d} \end{bmatrix}, \qquad r \in \{0,1\}, \text{ at random} \tag{9}$$

Relation 8 thus becomes:

$$\begin{aligned} \boldsymbol{x}^{o1} &= \left\{ r_{1,i} \cdot x_i^M + (1-r_{1,i}) \cdot x_i^F \right\} \\ \boldsymbol{x}^{o2} &= \left\{ r_{2,i} \cdot x_i^F + (1-r_{2,i}) \cdot x_i^M \right\} \end{aligned} \qquad i = 1, 2, ..., d \tag{10}$$

Intermediate crossover produces the most changes in the population. In this case, *r* is a 2x*d* matrix containing random values between 0 and 1, as described by relation (11). Relation 10 can be used to calculate the values of the offspring variables.

$$\boldsymbol{r} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,d} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,d} \end{bmatrix}, \qquad r \in [0;1], \text{ uniform and random} \tag{11}$$

In the case of *linear crossover*,e, $\boldsymbol{r}$ is a 2x1 matrix containing random values between 0 and 1.

$$\boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \qquad r \in [0;1], \text{ uniform and random} \tag{12}$$

Relation 8 thus becomes:

$$\begin{aligned} \boldsymbol{x}^{o1} &= \left\{ r_1 \cdot x_i^M + (1-r_1) \cdot x_i^F \right\} \\ \boldsymbol{x}^{o2} &= \left\{ r_2 \cdot x_i^F + (1-r_2) \cdot x_i^M \right\} \end{aligned} \qquad i = 1, 2, ..., d \tag{13}$$

2.5 Mutation

Mutation in EA mimics the natural process by changing the values of a small number of randomly selected genes. Mutation is subsequently applied after crossover and the authors refer to both as *reproduction*.

Mutation is performed according to the mutation rate μ, which represents the percentage of genes that are subjected to mutation. The mutation rate conventionally ranges between 0 and 1. Therefore, the number of mutated genes is $n_{mg} = \mu \cdot d \cdot n_c$.

In this paper, the mutation rate ranges from 0.05 to 0.5.

For real coded EA, mutation can be performed by replacing a randomly chosen gene from a randomly chosen chromosome with a randomly generated value in the function domain. This method is addressed to as *random mutation*. For any chromosome $\boldsymbol{x}_i$, the position of a gene to be mutated is considered ℓ_{mut}. The new value of the gene is calculated as follows:

$$\begin{aligned} &\boldsymbol{x}_{i,\ell mut} = (x_{max} - x_{min}) \cdot a + x_{min}, \quad i \in \{1,2,...,n_c\} \\ &a \in [0;1], \text{ uniform ant random} \end{aligned} \tag{14}$$

The authors have also tested another kind of mutation – variable step mutation. For any chromosome $\boldsymbol{x}_i$ and position ℓ_{mut} of a gene to be mutated, the new value of the gene is established according to the relation:

$$\begin{aligned} &x_{i,\ell mut} = x_{i,\ell mut} \pm (x_{max} - x_{min}) \cdot a \cdot step \\ &a \in [0;1], \text{ uniform at random} \end{aligned} \tag{15}$$

The value of a gene is altered by a quantity proportional to the domain. The variable step represents that proportion and it is divided by 10, after a number of iterations, to refine the current solution. The initial step size is 0.1. This particular mutation type may cause a variable to exceed the domain. If this case, the variable is assigned the value of the limit it has exceeded.

2.6 Elitism

Elitism implies that a small number of the most suitable solutions are copied unaltered to the next generation. The approach put forth in this paper copies only one solution (the best), as described by Eq. 16.

$$\begin{aligned} &f(\boldsymbol{x}_{elit}^t) = \min\{f(\boldsymbol{x}_i^t)\}, \quad i = 1,2,...,n_c \\ &\boldsymbol{x}_1^{t+1} = \boldsymbol{x}_{elit}^t \end{aligned} \tag{16}$$

2.7 Stopping criteria

Once a solution has been found, one has determine the extent to which this solution provides a fair answer to the problem. The best solution is usually a compromise between quality and computational effort. Consequently, EA performance has been analyzed under different conditions, considering the following two stopping criteria:

- the quality of the solution no longer improves after a certain number of iterations;
- the maximum number of generations is reached (backup criterion).

3 Results and Discussion

Based on the above algorithm, a software tool has been developed in MatLab 2012. This software tool allows the user to change the values of basic parameters, such as population size, crossover rate and mutation rate. Moreover, the user can change between the different selection, reproduction and mutation types, and can set specific parameters.

The EA settings are the following:

- the number of variables for each function is 10;
- truncation selection is used, with $\sigma = 0.5$ survival rate (as a result, the best individuals are selected in a reduced computing time);
- the intermediate crossover method is used, which ensures a high population diversity is assured;
- both mutation methods are tested. In the case of variable step mutation, the step decreases every 200 iterations;
- the maximum iteration number is 5000;
- for each situation, the algorithm runs 20 times.

The influence of the parameters is analyzed for the following values:

- population size: $n_c = \{20; 40; 60; 80; 100\}$;
- crossover rate: $\chi = \{0.5; 0.6; 0.7; 0.8; 0.9\}$;
- mutation rate: $\mu = \{0.01; 0.05; 0.1; 0.25; 0.5\}$.

Once the optimal parameters have been established for each function, the provided values are presented for different configurations of genetic operators. The cases under discussion are summarized in Table 1. When not subjected to analysis, the size of the population counts 20 individuals, and the crossover probability is 0.8. Mutation probability is set after running a test.

Table 1
Cases under discussion for EA testing

	EA1	EA2	EA3	EA4	EA5
Selection:	Truncation	Tournament	Ranking	Truncation	Truncation
• Survival rate:	$\sigma = 0.5$	-	-	$\sigma = 0.5$	$\sigma = 0.5$
• Number of competitors:	-	$n_{comp} = n_c/4$	-	-	-
Crossover	Intermediate	Intermediate	Intermediate	Discrete	Linear

Three mathematical test functions have been implemented within the software tool: Schwefel, Rastrigin and Rosenbrock. The definitions of these functions, the intervals of their variables and their three-dimensional representation are presented below.

3.1 Rastrigin function

The Rastrigin function is a nonlinear function (relation (17)). It is based on De Jong function, additionally including the cosine term, which periodically generates local minimum values with regulated distribution [21]. Due to the great number of local minimum values, the optimization EAs have difficulties in finding the global minimum [23]. The corresponding 3D plot is presented in Fig. 3. The global optimum value is recorded for $x = 0$ ($f(x) = 0$).

$$f = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi \cdot x_i)), \quad i = \overline{1,n}\ ; \quad -5.12 \le x_i \le 5.12 \tag{17}$$

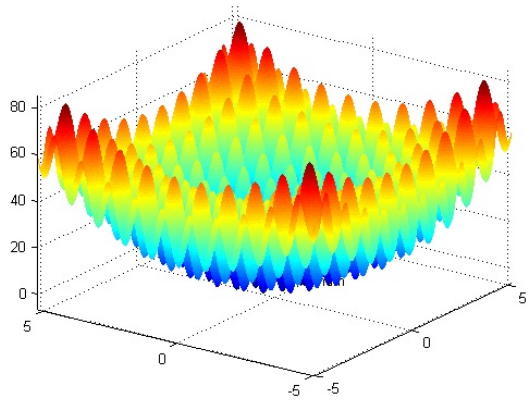


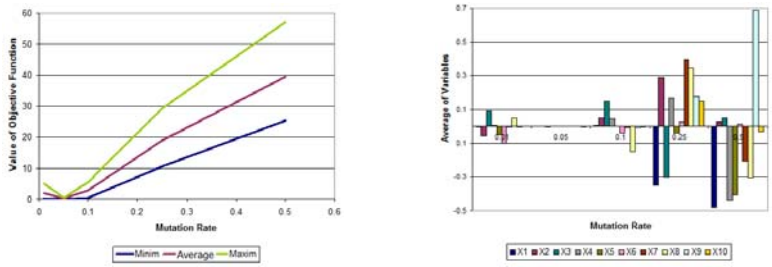
Figure 3
3D Rastrigin's plot

The evolution of minimum, average and maximum function values are presented in Fig. 4.a. The average variable values are presented in Fig. 4.b. Random mutation for a 0.05 rate constantly leads to very good results. Variable step mutation produces inadequate results (Fig. 5), even for reduced mutation rates.

In what follows, a 0.05 mutation rate will be used.

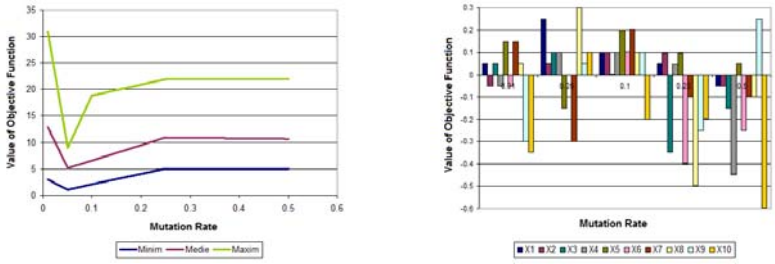
Fig. 6 describes the influence of the crossover rate. For a 0.7 or a 0.8 rate, the solutions are concentrated in the neighbourhood of the global minimum. Fig. 7 illustrates the population influence. It should be emphasized that a large population size has a negative effect on the quality of the solution.

The numerical results corresponding to the Rastrigin function, algorithms EA1-EA5 (Table 1), are summarized in Table 2. The settings used in the analyses are: population size $n_c = 20$; crossover rate $\chi = 0.8$; mutation type – random; mutation rate $\mu = 0.05$.



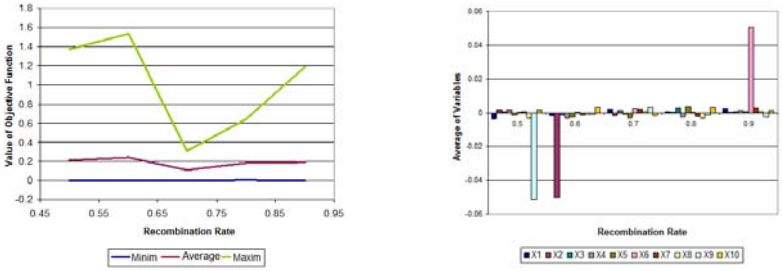
a) b)

Figure 4
Mutation rate influence – random mutation; a) function value; b) average variables' values



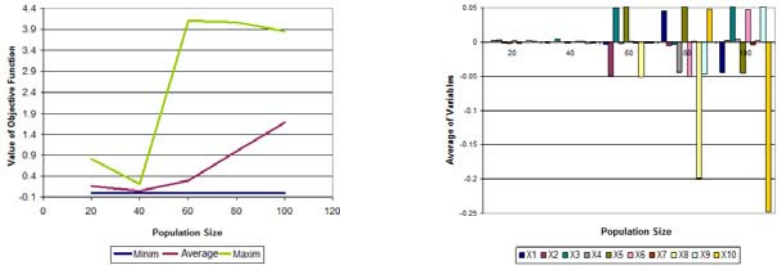
a) b)

Figure 5
Mutation rate influence – variable step mutation; a) function value; b) average variables' values



a) b)

Figure 6
Crossover rate influence a) function value; b) average variables' values



a) b)

Figure 7
Population size influence a) function value; b) average variables' values

Table 2
Rastrigin function numerical results' synthesis

	EA1	EA2	EA3	EA4	EA5
$f(\mathbf{x}_i)$ average value	0.1509	0.1714	0.5227	0.2596	0.2847
$f(\mathbf{x}_i)$ minimum value	0.0094	0.0012	0.0221	0.0147	0.0039
x_1 average value	-0.0002	0.0000	0.0035	0.0000	0.0035
x_2 average value	0.0000	0.0029	-0.0001	0.0029	-0.0001
x_3 average value	-0.0009	0.0004	-0.0022	0.0004	-0.0022
x_4 average value	-0.0006	-0.0016	0.0392	-0.0016	0.0392
x_5 average value	-0.0002	-0.0015	-0.0189	-0.0015	-0.0189
x_6 average value	0.0004	-0.0012	-0.0000	-0.0012	-0.0000
x_7 average value	0.0007	-0.0001	-0.0018	-0.0001	-0.0018
x_8 average value	0.0017	0.0004	-0.0204	0.0004	-0.0204
x_9 average value	-0.0005	-0.0007	0.0189	-0.0007	0.0189
x_{10} average value	0.0011	0.0000	-0.0008	0.0000	-0.0008

These results are graphically presented in Fig. 8.

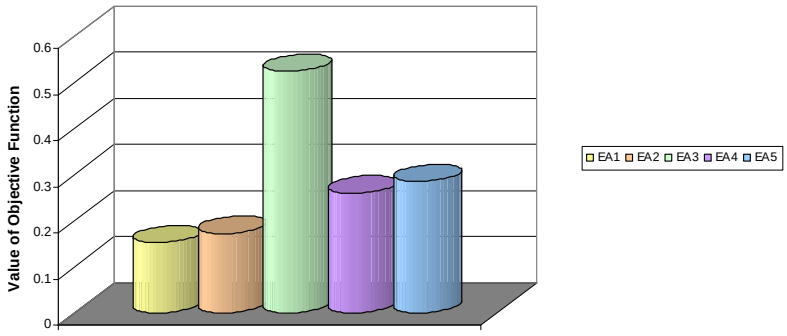


Figure 8
Rastrigin function values in case of each algorithm EA1-EA5

Fig. 8 and Table 2 highlight the following aspects:

- EA1 produces the best results (truncation selection and intermediate crossover);
- the average function values range from 0.1509 to 0.5227;
- the average variable values are range from -0.0204 to 0.0392; both limits are recorded for EA3 (ranking selection and intermediate crossover);
- truncation selection induces high quality behaviour, if associated with powerful crossover methods;
- compared to the EA3, other combinations lead to insignificant variations when the number of variables is increased.

3.2 Schwefel function

The Schwefel function is a nonlinear function (relation (18)). This function in characterized by an increased distance between the global minimum and the subsequent one [21], could lead to a wrong direction of the EA [23]. The corresponding 3D plot is presented in Fig. 9. The global optimum value is recorded for $x = 420.9687$ ($f(x) = 0$).

$$f = 418,9829 \cdot n - \sum_{i=1}^{n} [-x_i \cdot \sin(\sqrt{|x_1|}), \quad i = \overline{1,n}\ ; \quad -500 \le x_i \le 500 \tag{18}$$

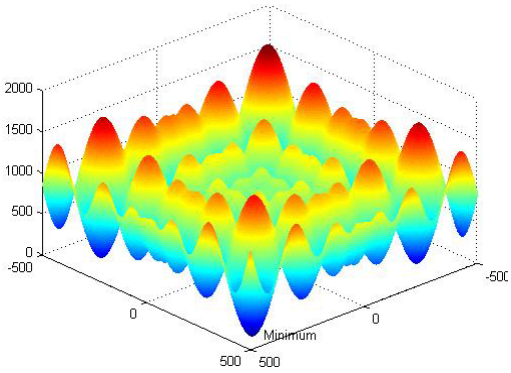
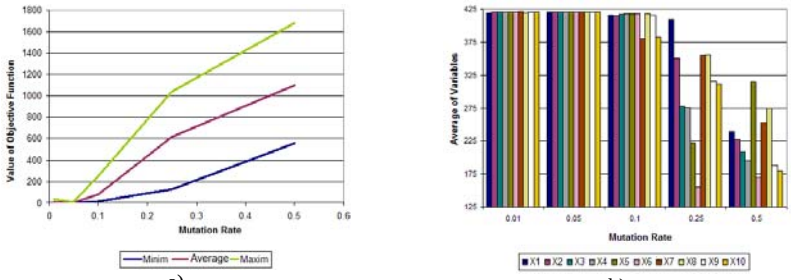


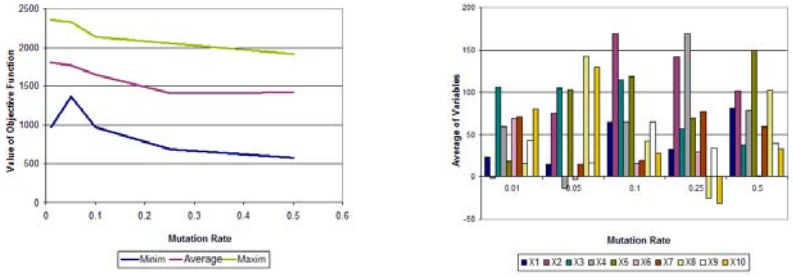
Figure 9
3D Schwefel's plot

The evolution of minimum, average and maximum function values are presented in Fig. 10.a. The average variable values are presented in Fig. 10.b. Random mutation for a 0.05 rate constantly leads to very good results.



a) b)

Figure 10
Mutation rate influence – random mutation; a) function value; b) average variables' values

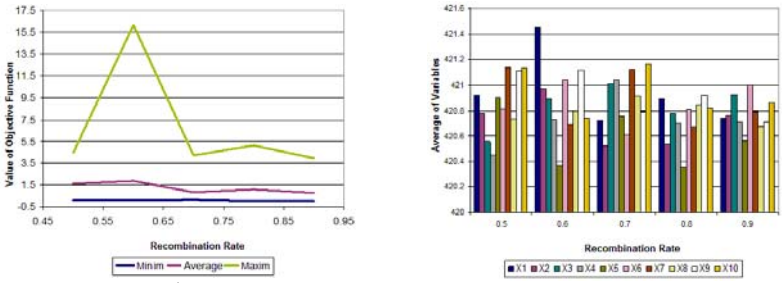


a) b)

Figure 11
Mutation rate influence – variable step mutation; a) function value; b) average variables' values

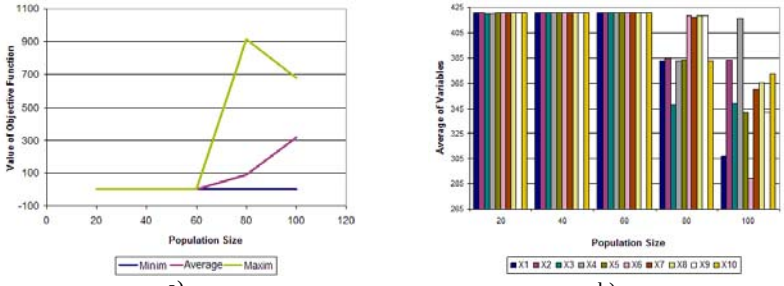
In what follows, a 0.05 mutation rate will be used.

Fig. 12 describes the influence of the crossover rate. The solution quality is improving for high crossover rate values. Fig. 13 illustrates the population influence. It should be emphasized that a large population size has a negative effect on the quality of the solution.



a) b)

Figure 12
Crossover rate influence: a) function value; b) average variables' values



a) b)

Figure 13
Population size influence: a) function value; b) average variables' values

The numerical results corresponding to the Schwefel function, algorithms EA1-EA5 (Table 1) are summarized in Table 3. The settings used in the analyses are: population size $n_c = 20$; crossover rate $\chi = 0.7$; mutation type – random; mutation rate $\mu = 0.05$.

Table 3
Schwefel function numerical results' synthesis

	EA1	EA2	EA3	EA4	EA5
$f(\mathbf{x}_i)$ average value	0.7936	0.6936	22.6335	1.7793	1.6594
$f(\mathbf{x}_i)$ minimum value	0.0293	0.0053	0.8711	0.1681	0.0281
x_1 average value	420.82	420.88	418.85	420.80	420.96
x_2 average value	421.02	420.86	418.46	420.82	420.78
x_3 average value	420.74	420.92	419.34	420.84	420.60
x_4 average value	420.94	420.64	417.94	421.00	420.66
x_5 average value	420.70	421.00	418.96	420.90	420.68
x_6 average value	420.90	420.98	419.18	420.76	420.90
x_7 average value	420.80	420.82	418.74	421.02	420.98
x_8 average value	421.00	421.08	417.68	420.82	420.78
x_9 average value	420.86	420.96	418.78	421.18	420.66
x_{10} average value	420.87	420.87	418.00	421.09	420.67

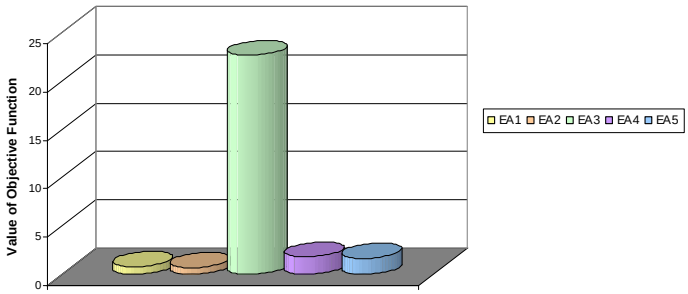


Figure 14
Schwefel function values in case of each algorithm EA1-EA5

Fig. 14 and Table 3 highlight the following aspects:

- EA2 produces the best results (tournament selection and intermediate crossover);
- the average function values range from 0.6936 to 22.6335;
- the average variable values are range from 420.64 to 421.08; both limits are recorded for EA3 (ranking selection and intermediate crossover);
- truncation selection and tournament selection induce high quality behaviour, if associated with powerful crossover methods;
- other combinations lead to insignificant variations when the number of variables is increased.

3.3 Rosenbrock function

Rosenbrock function (relation (19)) represents a classical optimization problem. The global optimum is situated inside a long and narrow valley. The searching algorithms reache the valley easily, but the convergence to the global optimum is difficult. This function is used fortest test the performance of the searching algorithm [21]. The corresponding 3D plot is presented in Fig. 15. The global optimum value is recorded for $x = 1$, $f(x) = 0$.

$$f = \sum_{i=1}^{n-1} [100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \; ; \qquad -2{,}048 \le x_i \le 2{,}048 \tag{19}$$

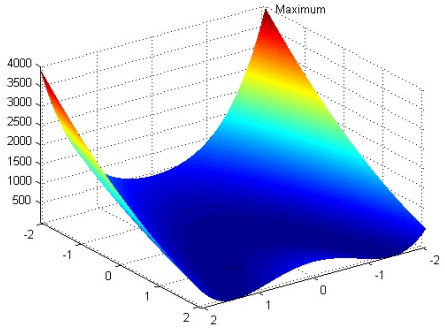
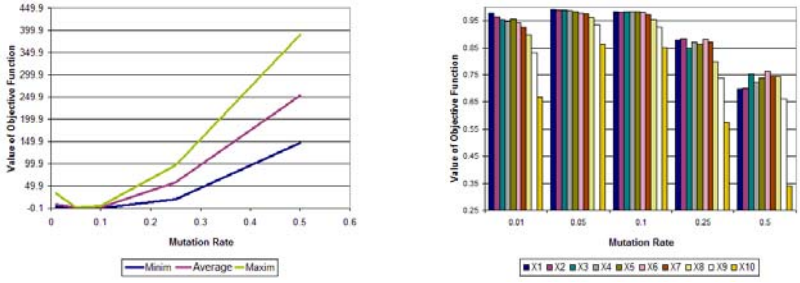


Figure 15
Rosenbrock function 3D plot

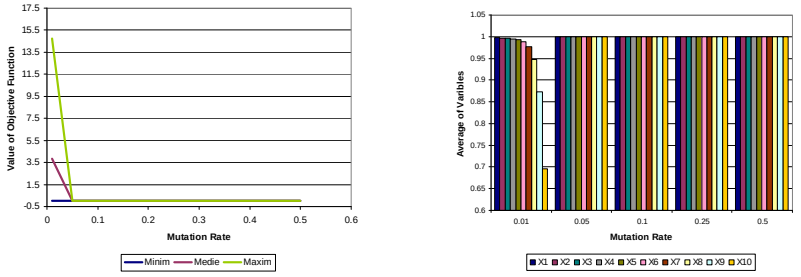
The variable step mutation proves to be the most suitable mutation method for determining the global minimum of this function (Fig. 16).



a) b)

Figure 16
Mutation rate influence – random mutation; a) function value; b) average variables' values

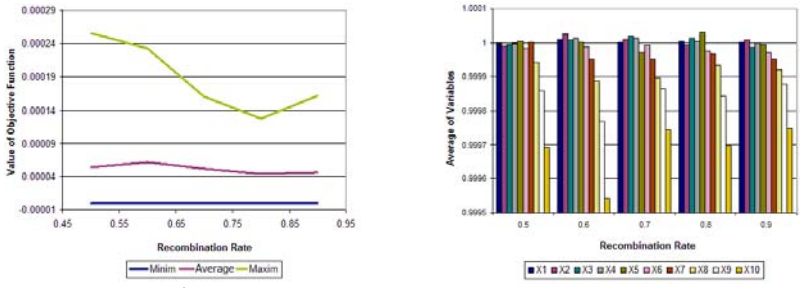
Variable step mutation with mutation rates greater than 0.05 leads to the right solution, as seen in Fig. 17. In what follows, a 0.25 mutation rate will be used.



a) b)

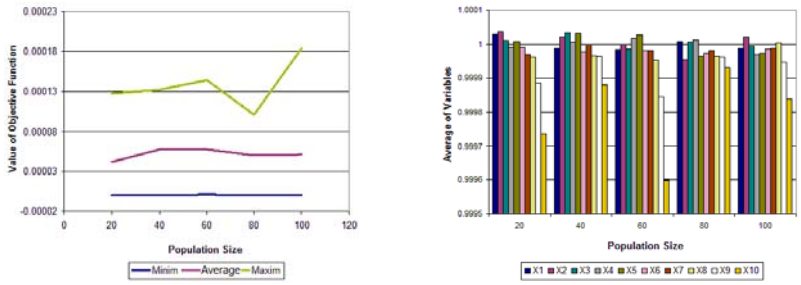
Figure 17
Mutation rate influence – random mutation; a) function value; b) average variables' values

Fig. 18 describes the influence of the crossover rate. Fig. 19 illustrates the population influence. It should be emphasized that their influence is not significant on the quality of the solution.



a) b)

Figure 18
Crossover rate influence: a) function value; b) average variables' values



a) b)

Figure 19
Population size influence: a) function value; b) average variables' values

The numerical results corresponding to the Rosenbrock function, algorithms EA1-EA5 are summarized in Table 4. The settings used in the analyses are: population size $n_c = 20$; crossover rate $\chi = 0.8$; mutation type – variable step; mutation rate $\mu = 0.25$.

Table 4
Rosenbrock numerical results' synthesis

	EA1	EA2	EA3	EA4	EA5
$f(\mathbf{x}_i)$ average value	0.0001	0.0001	0.0000	0.0001	0.0001
$f(\mathbf{x}_i)$ minimum value	0	0	0	0	0
x_1 average value	1.0000	1.0001	1.0000	1.0000	0.9999
x_2 average value	1.0000	1.0000	1.0000	0.9999	1.0000
x_3 average value	1.0000	1.0000	1.0000	0.9999	0.9999
x_4 average value	1.0000	1.0000	0.9999	1.0000	0.9999
x_5 average value	1.0000	1.0000	0.9999	0.9999	0.9999
x_6 average value	0.9999	0.9999	0.9999	0.9999	0.9999
x_7 average value	1.0000	0.9999	0.9999	0.9999	0.9999
x_8 average value	0.9999	0.9999	0.9999	0.9999	0.9999
x_9 average value	0.9999	0.9999	0.9999	0.9998	0.9999
x_{10} average value	0.9999	0.9998	0.9999	0.9998	0.9998

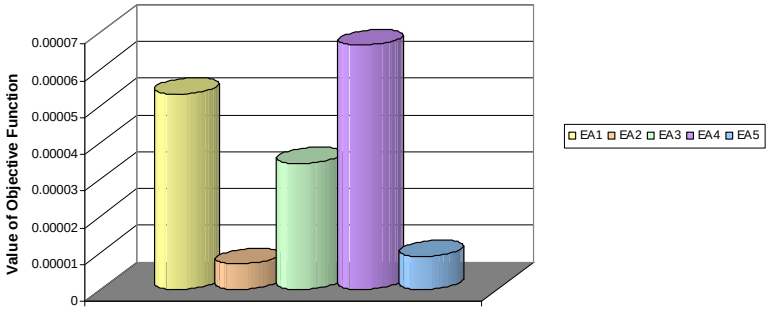


Figure 20
Rosenbrock function values in case of each algorithm EA1-EA5

Fig. 20 and Table 4 highlight the following aspects:

- EA2 produces the best results (tournament selection, intermediate crossover);
- the average function values range from 0.0000 to 0.0001;
- the average variable values are range from 0.9998 to 1.0001;

- EA2 and EA5 algorithms are constantly producing very good results;
- variable step mutation having high mutation rate could be considered the main mechanism.

3.4 A contrastive analysis of the three functions

Table 5 shows the best parameter values for the three functions discussed above.

Table 5
Results and settings for 20, 50 and 100 variables

Function	Variables	Population size	Mutation type	Mutation rate	Crossover rate	Function value	Variables' average value
Rastrigin	20	50	Random	0.01	0.8	0.0029	-0.0005
Rastrigin	50	100	Random	0.01	0.8	0.0170	0.0001
Rastrigin	100	200	Random	0.01	0.8	2.9807	0.0005
Schwefel	20	20	Random	0.01	0.7	0.0074	420.9936
Schwefel	50	100	Random	0.01	0.7	0.0039	420.9975
Schwefel	100	200	Random	0.01	0.7	1.7627	420.9068
Rosenbrock	20	20	Variable step	0.25	0.8	0.0000	0.9999
Rosenbrock	50	50	Variable step	0.25	0.8	0.0003	0.9999
Rosenbrock	100	100	Variable step	0.25	0.8	0.0158	0.9996

The computing process stops if during 200 iterations the result does not improve or the maximum number of iterations is reached (in this case – 5000 iterations). Truncation selection produces good results, characterized by reduced computational effort. A 0.5 survival rate has been used. Intermediate crossover produces the greatest population diversity.

Conclusion

The present research is not only necessary, but also extremely useful step to further developments in EA based OPF computing and TNEP analysis.

In this paper the authors have attempted to discuss the behaviour of the EAs in several scenarios. A suitable software tool has been developed for the EAs benchmark. Different combinations between the genetic operators and their values have been analyzed for several number of variables. The results can be applied to power systems analysis.

The values of the EA parameters are directly influenced by the nature of the optimization problem and by the number of variables.

For a small number of variables, ranking selection leads to very good results.

It is recommended that the population size should not to be too large.

High rate values are recommended for crossover, while reduced rate values for random mutation. If variable step mutation is used, high mutation rate values are can be employed (0.05-0.5). A 0.25 value has been adopted for the analyzed cases. This strategy leads to a decrease in population size. Such an algorithm is very close to the evolutionary computing strategy.

Variable step mutation offers the best results for Rosenbrock function, when used with high probability. Rastrigin and Schwefel functions can be solved using random mutation and a low mutation rate.

If variable step mutation is employed, the influence of the crossover rate and of the population size is less significant; the algorithm is mutation driven.

Satisfactory results have been obtained for a bigger number of variables

Acknowledgement

This work was supported by the strategic grant POSDRU/89/1.5/S/57649, Project ID 57649 (PERFORM-ERA), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013 and by the strategic grant POSDRU/ 88/1.5/S/50783 (2009) of the Ministry of Labour, Family and Social Protection, Romania, co-financed by the European Social Fund – Investing in people.

References

[1] Weise T., Global optimization algorithms – theory and application, 2nd Edition, http://www.it-weise.de/, 2010;

[2] Chung T.S., Li Y.Z., A Hybrid GA Approach for OPF with Consideration of FACTS Devices, IEEE Power Engineering Review, vol.21, no.2, 2001, pp.47-50;

[3] Bakirtzis A. G., Biskas P. N., Zoumas C. E., Petridis V., Optimal Power Flow by Enhanced Genetic Algorithm, IEEE Power Engineering Review, vol.22, no.2, 2002, pp.60-66;

[4] Todorovski M., Rajicic D., An initialization procedure in solving optimal power flow by genetic algorithm, IEEE Transactions on Power Systems, vol.21, no.2, 2006, pp.480-487;

[5] Chan K.Y., Ling S.H., Chan K.W., Iu H.H.C., Pong G.T.Y., Solving multi-contingency transient stability constrained optimal power flow problems with an improved GA, IEEE Congress on Evolutionary Computation, CEC 2007, pp.2901-2908;

[6] Malik I.M., Srinivasan D., Optimum power flow using flexible genetic algorithm model in practical power systems, 9th International Power and Energy Conference IPEC 2010, pp.1146-1151;

[7] Rahul J., Sharma Y., Birla D., A New Attempt to Optimize Optimal Power Flow Based Transmission Losses Using Genetic Algorithm, 4th International Conference on Computational Intelligence and Communication Networks, CICN 2012, pp.566-570;

[8] Kazemi A., Jalilzadeh S., Mahdavi M., Haddadian H., Genetic algorithm-based investigation of load growth factor effect on the network loss in TNEP, 3rd IEEE Conference on Industrial Electronics and Applications ICIEA 2008, pp.764-769;

[9] Jalilzadeh S., Kazemi A., Mahdavi M., Haddadian H., TNEP considering voltage level, network losses and number of bundle lines using GA, 3rd International Conference Electric Utility Deregulation and Restructuring and Power Technologies DRPT 2009, pp.1580-1585;

[10] Maghouli P., Hosseini S.H., Buygi M.O., Shahidehpour M., A Multi-Objective Framework for Transmission Expansion Planning in Deregulated Environments, IEEE Transactions on Power Systems, vol.24, no.2, 2009, pp.1051-1061;
[11] Maghouli P., Hosseini S.H., Buygi M.O., Shahidehpour M., A Scenario-Based Multi-Objective Model for Multi-Stage Transmission Expansion Planning, IEEE Transactions on Power Systems, vol.26, no.1, 2011, pp.470-478;
[12] Rodriguez J., Falcao D.M., Taranto G.N., Almeida H., Short-Term Transmission Expansion Planning by a Combined Genetic Algorithm and Hill-Climbing Technique, 15th International Conference on Intelligent System Applications to Power Systems, ISAP 2009, pp.1-6;
[13] F. Cadini, E. Zio, C.A. Petrescu, Optimal expansion of an existing electrical power transmission network by multi-objective genetic algorithms, Reliability Engineering & System Safety, Volume 95, Issue 3, 2010, pp. 173-181;
[14] Huang Wei, Feng Li, He Zijun, Cui Junzhao, Zhang Li, Transmission network planning with N-1 security criterion based on improved multi-objective genetic algorithm, 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies DRPT 2011, pp.1250-1254;
[15] Asadzadeh V., Golkar M.A., Moghaddas-Tafreshi S.M., Economics-based transmission expansion planning in restructured power systems using decimal codification genetic algorithm, IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies AEECT 2011, pp.1-8;
[16] Sirikum J., Techanitisawad A., Kachitvichyanukul V., A New Efficient GA-Benders' Decomposition Method: For Power Generation Expansion Planning With Emission Controls, IEEE Transactions on Power Systems, vol.22, no.3, 2007, pp.1092-1100;
[17] Murugan P., Kannan S., Baskar S., Application of NSGA-II Algorithm to Single-Objective Transmission Constrained Generation Expansion Planning, IEEE Transactions on Power Systems, vol.24, no.4, 2009, pp.1790-1797;
[18] Kannan S., Baskar S., McCalley J.D., Murugan P., Application of NSGA-II Algorithm to Generation Expansion Planning, IEEE Transactions on Power Systems, vol.24, no.1, 2009, pp.454-461;
[19] Wang D.T., Ochoa L.F., Harrison G.P., Expansion planning of distribution networks considering uncertainties, Proceedings of the 44th International Universities Power Engineering Conference UPEC 2009, pp.1-5;
[20] Wang D.T., Ochoa L.F., Harrison G.P., Modified GA and data envelopment analysis for distribution network expansion planning under uncertainty, IEEE Transactions on Power Systems, vol.26, no.2, 2011, pp.897-904;
[21] Molga M., Test functions for optimization needs, http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, 2005;
[22] Haupt R.L., Haupt S.E., Practical genetic alghorithms, 2nd Edition, John Wiley & Sons, 2004;
[23] Digalakis J.G., Margaritis K.G., An experimental study of benchmarking functions for genetic algorithms, IEEE International Conference on Systems, Management and Cybernetics, 2000 , vol.5, pp.3810-3815.