# Adaptation of an Evolutionary Algorithm in Modeling Electric Circuits

J. Hájek

*This paper describes the influence of setting control parameters of a differential evolutionary algorithm (DE) and the influence of adapting these parameters on the simulation of electric circuits and their components. Various DE algorithm strategies are investigated, and also the influence of adapting the controlling parameters (Cr, F) during simulation and the effect of sample size. Optimizing an equivalent circuit diagram is chosen as a test task. Several strategies and settings of a DE algorithm are evaluated according to their convergence to the right solution.*

*Keywords: Differential evolution, adaptation, Maple, modeling of electric circuits.*

## 1 Introduction

Many cases of the use of evolutionary algorithms in optimizing technical tasks have been published, most of them in the design of antennas, microstrip lines and other microwave elements. These tasks are characterized by difficult geometry, which can be solved only numerically. A solution based on an analytic description followed by optimizations is almost impossible. Therefore, stochastic or evolutionary algorithms are used for such tasks.

In the electrical engineering industry, synthesis of high-frequency circuits (for example radio-interference filters) is a very frequent task. The goal is to find values of electrical components for which the circuit behaves according to our requirements. This can also be called optimization; the target function is for example the difference between real behavior of the circuit and its scheduled frequency response. As shown in this paper, an evolutionary algorithm can also be used in such tasks.

A big advantage when using evolutionary algorithms is their robustness and the large probability of finding a right solution or more than one right solution. The main disadvantage is the longer time needed to finalize the computation (especially when optimizing several variables simultaneously) and great sensitivity to the control settings. Therefore, some studies have recently been undertaken to eliminate these unwanted features. One way is to use auto-adaptation of the controlling parameters during the optimization process. [1]

There are some special features of using an evolutionary algorithm when optimizing electric circuits: first, there are a huge range of variables from pico ($10^{-12}$) to units or tens of mega ($10^7$). Next, there is a huge sensitivity to a small variance in inputs. A resonance effect in *LC* circuits (consisting only from inductors *L* and capacitors *C*) can serve as an example. The final difficulty is the huge dimension of the area of the solution. Most simple digital or *LC* filters are composed of at least 3–4 elements, but it is necessary to consider parasitic effects and initial conditions. The final number of wanted variables can be 10 or more. Thus these engineering tasks are more difficult than ordinary mathematical tests for optimizing algorithms. Some tests (De Jong, Rastring, Schwefel and Ackley's function) are introduced in [2].

## 2 Optimization using *DE*

In technical practice, the differential evolutionary algorithm is commonly considered as a very reliable and robust optimizing technique. *DE* was developed for computing with real numbers and the domain of definition can be specified. (When solving box constrained problems) *DE* works with a population of solutions *P*, the size (dimension) of which is *NP* individuals. Population *P* changes during *G* generations. Each individual from the current population can be presented as *D*-dimensional vector, when *D* presents the dimension of the definition domain. *D* corresponds to the number of required variables. *DE* can be described in pseudo-code near to PASCAL as follows:

```
1. create initial population $P = (x_1, x_2, \ldots, x_N)$; (randomly)
2. for $i := 1$ to $G$ do
3.     for $j := 1$ to $N$ do
4.         create mutation vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_D)$
5.         create new solution $y_j$ by combination of
           "parents" $u$ and $x_j$
6.         if $f(y_j) < f(x_j)$ then replace $x_j$ by $y_j$; ($f$ means
           target function)
7.         end if
8.     end for
9. end for;  (or cycle repeat – until with suitable terminating
   condition)
```

This process is essentially common for all evolutionary algorithms, the core of *DE* is built-in in the strategy of creating mutation vector $\boldsymbol{u}$. There are several ways (strategies) for generating mutation vector $\boldsymbol{u}$. The most frequent strategy is called *DE/rand*, when the weighted difference from randomly chosen solutions (individuals from *P*) is used:

$$\boldsymbol{u} = r_1 + F(r_2 - r_3). \tag{1}$$

The disparity $r_1 \neq r_2 \neq r_3 \neq x_j$ must be valid and $r_i$ is randomly chosen from population *P*. The name of this strategy also results from this equation (*rand*), as does the name of the algorithm (weighted **d**ifference → ***DE***). Control factor *F* is entered by the user and the authors of *DE* recommend $F = 0.8$. Another strategy, known as *DE/best*, generates mutation vector $\boldsymbol{u}$ as follows:

$$\boldsymbol{u} = x_{\min} + F(r_1 - r_2). \qquad (2)$$

In this case, the disparity $r_1 \neq r_2 \neq r_3 \neq x_{\min}$ is again valid and the value $x_{\min}$ represents the best solution in previous and present populations, i.e. it represents the best solution with minimal target function ($f$). This explains the name of the strategy (*best*). After creating a mutation (sometimes called a "noise vector") it is possible to make descendant $y$ from "parents" $x_j$ and $\boldsymbol{u}$. Elements $y_k$ for $k = 1, 2, \ldots, D$ are created according to the following equations:

$$y_k = u_k \text{, while } U_k \le Cr \text{ or } l = k \text{ (element } y_k \text{ changes)}, \qquad (3)$$

$$y_k = x_{jk} \text{, while } U_k > Cr \text{ and } l \neq k \text{ (element } y_k \text{ does not change)}. \qquad (4)$$

Variable $l$ is a randomly chosen number from the set {1, 2, …, $D$}, random variables $U_1$ up to $U_D$ are from the interval [0, 1] with normal distribution. *Cr* is the next control factor that influences the frequency of crossing; $Cr \in [0,1]$. The authors recommend us to choose $Cr = 0.5$; i.e. 50 % probability of changing $x_{jk}$. It is recommended to choose $NP = 10\,D$. The advantage of *DE* is its simplicity, because the introduced procedure is simply programmable. The settings of factors *F*/*Cr* and a suitable choice of the population size are the biggest disadvantages. Convergence of *DE* depends dramatically on the controlling factors. Values $F = 0.8$ and $Cr = 0.5$ do not always guarantee good convergence. Various settings are suitable in various cases. Auto-adaptation of algorithm *DE* can be used as a defense against this unwanted sensitivity. It can be provided by changing weight factor *F* according transient results, as shown in [1]:

$$F = \max\left(F_{\min}, 1 - \left|\frac{f_{\max}}{f_{\min}}\right|\right) \text{, while } \left|\frac{f_{\max}}{f_{\min}}\right| < 1, \qquad (5)$$

$$F = \max\left(F_{\min}, 1 - \left|\frac{f_{\min}}{f_{\max}}\right|\right) \text{, while } \left|\frac{f_{\max}}{f_{\min}}\right| \ge 1. \qquad (6)$$

Another way, which was not tested in this paper is to insert factors *F* and *Cr* as variables directly into the population *P* and keep them developing in each generation. Successful values will survive longer and can affect the optimization process positively. This way is described in details in [2].

## 3 Description of test task

Optimizing a model of a suppressor capacitor was chosen as a test task for the *DE* algorithm. This model has four input variables $R_1$, $R_2$, $L_1$, $C_1$, which affect its behavior, especially insertion loss. The output variable of the model is the computed insertion loss dependence. The goal of the optimization process was to find out values of the input variables ($R_1$, $R_2$, $L_1$, $C_1$), such that the model has the same dependence of the insertion loss as shown on the Fig. 1. The target function for the optimizing routine was defined as the sum of square of the variations between model insertion loss and the measured insertion loss. A common foil capacitor with plastic dielectric and axial outlets (TC 218 100 nF/630 V) served as an object for measuring. The insertion loss dependence (i.e. the amplitude frequency response) of capacitor TC 218 was measured using signal analyzer PMM 9010 in accordance with standard CISPR 17 [3].

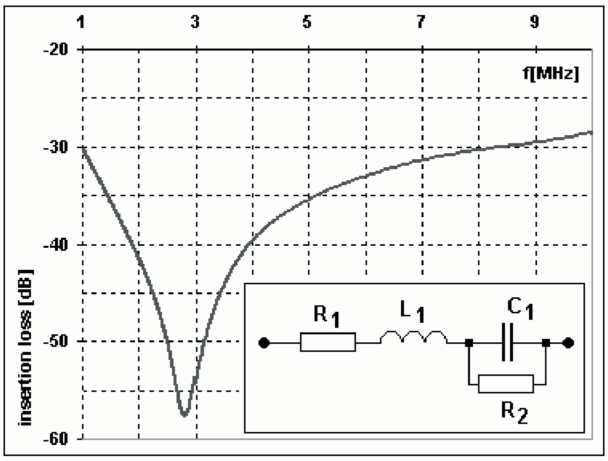

Fig. 1: Model of the capacitor (below, right) and the insertion loss measured on real capacitor TC 218

The domain of the solution has a dimension of 4, because there are four wanted elements $R_1$, $R_2$, $L_1$, $C_1$. This domain was restricted to real ranges: $R_1(\Omega) \in [0.01;\, 1]$, $R_2(\Omega) \in [10^4;\, 10^7]$, $C_1(\text{nF}) \in [80;\, 120]$, $L_1(\text{nH}) \in [0;\, 60]$. This restriction has two meanings: First, it accelerates computation (smaller space, which should be sought through) and second, it separates solutions that are mathematically correct, but practically impossible.

Another necessary procedure was to scale all variables before running the algorithm. This was necessary because of the potential influence of round-off errors. The values of the variables in the test task range from Pico farads to tens of MHz. After finishing *DE*, all results were re-scaled back into common units. The following transformations were used for scaling and re-scaling:

$$R_{i\,\text{norm}} = \frac{R_i}{R_N}, \qquad (7)$$

$$C_{1\,\text{norm}} = C_1 2\pi f_N R_N, \qquad (8)$$

$$L_{1\,\text{norm}} = L_1 2\pi \frac{f_N}{R_N}. \qquad (9)$$

The reference values were $R_N = 1\,\Omega$ and $f_N = 1\,\text{MHz}$. All calculations were carried out in absolute values. Relative units (dB) were used only in results and graphs. This was done to avoid the possibility of round-off errors.

All the optimization routines of the *DE* algorithm were written in a Maple worksheet. A model of a capacitor was implemented in Maple 9.5 with help from the SYRUP library [4]. The SYRUP library enables us to insert electric circuits directly into the Maple environment. The test task ran on a PC with the following configuration: Intel Celeron 2,4 GHz, RAM 256 MB, Windows 2000, Maple 9.5 with SYRUP library.

## 4 Results of simulations

All the tested settings of the *DE* algorithm are shown in Table 1. Each setting was checked out and tested ten times. Average results are introduced in Table 1. Individual calculations were done independently with a random start-up population. The evolution cycle ended in all tests after the 100th generation. This number of generations is sufficient in a relatively simple case such as this. Populations with ranges $2D$, $4D$, $10D$ and $16D$ were chosen, so that various strategies and convergences could be discussed. Weight factor $F$ was also simultaneously changed. This factor is responsible for creating mutation vector $\boldsymbol{u}$. At the beginning of the evolution, this $F$ factor was always set to 0.9. This value ensures outstanding mutations at the beginning of the evolution and fine-tuning of founded solution at the end.

The tested algorithms finished properly and discovered solutions in specified ranges. The final values of the calculations after re-scaling and averaging are shown in Table 2. It should be noted that more different values of $R_1$, $R_2$, $L_1$, $C_1$ can ensure right behavior of the insertion loss (output variable of the model). An infinite number of electric circuits can theoretically even have nearly the same characteristics. Another result from Table 2 is that the model of the capacitor can be simplified. Input variable $R_2$ (varying in a large range) has practically neutral influence on the characteristic of the circuit. The conformity with measured dependence is presented in Table 2 as the absolute peak error. The typical accuracy of the insertion loss measurement required by [3] is ±3 dB. The simple four-element model from Fig. 1 cannot provide greater conformity.

An evaluation and a comparison of various *DE* algorithm strategies was declared as main goal of this paper. This comparison can be based on an analysis of the convergence. The results of the first four tests are shown on the Fig. 2. There are

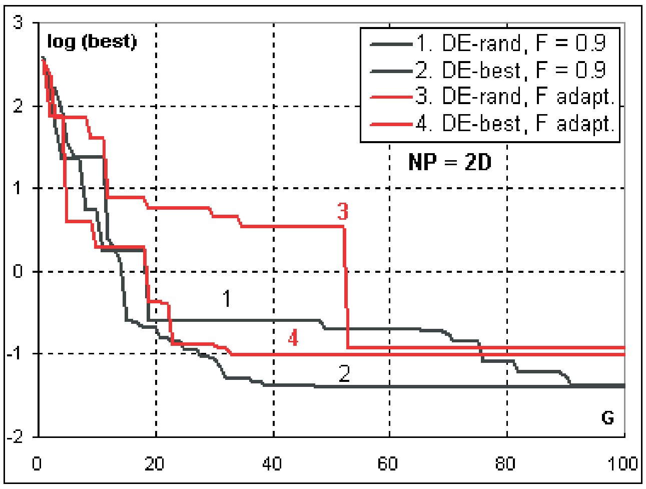

Fig. 2: Development of convergence in population $NP = 2D$

Table 1: Tested variants of *DE*. *NP* – size of population, *G* - number of generations, auto-adapt. – fixed or changed *F* factor, strategy – used strategy, $F_{100}$ – average weight factor *F* in the 100th generation, *Cr* – fixed factor, *best* – average minimum of target function, CPU time – average time needed to compute one test.

| No. test | *NP* | *G* | auto-adapt. | strategy | $F_{100}$ | *Cr* | *best* | CPU time (s) |
|---|---|---|---|---|---|---|---|---|
| 1. | 8 | 100 | No | rand | 0.9000 | 0.85 | 0.0420 | 39 |
| 2. | 8 | 100 | No | best | 0.9000 | 0.85 | 0.0400 | 34 |
| 3. | 8 | 100 | Yes | rand | 0.9427 | 0.85 | 0.1159 | 34 |
| 4. | 8 | 100 | Yes | best | 0.5000 | 0.85 | 0.0951 | 32 |
| 5. | 16 | 100 | No | rand | 0.9000 | 0.85 | 0.0488 | 70 |
| 6. | 16 | 100 | No | best | 0.9000 | 0.85 | 0.0400 | 70 |
| 7. | 16 | 100 | Yes | rand | 0.9715 | 0.85 | 0.0468 | 73 |
| 8. | 16 | 100 | Yes | best | 0.9824 | 0.85 | 0.0401 | 72 |
| 9. | 40 | 100 | No | rand | 0.9000 | 0.85 | 0.0656 | 216 |
| 10. | 40 | 100 | No | best | 0.9000 | 0.85 | 0.0400 | 214 |
| 11. | 40 | 100 | Yes | rand | 0.9994 | 0.85 | 0.0431 | 220 |
| 12. | 40 | 100 | Yes | best | 0.5000 | 0.85 | 0.0400 | 208 |
| 13. | 64 | 100 | No | rand | 0.9000 | 0.85 | 0.0476 | 385 |
| 14. | 64 | 100 | No | best | 0.9000 | 0.85 | 0.0400 | 380 |
| 15. | 64 | 100 | Yes | rand | 0.9992 | 0.85 | 0.0519 | 401 |
| 16. | 64 | 100 | Yes | best | 0.5000 | 0.85 | 0.0400 | 395 |

Table 2: Results of all variants after re-scaling and averaging

| No. test | $R_1$ (mΩ) | $R_2$ (MΩ) | $L_1$ (nH) | $C_1$ (nF) | Abs. peak error (dB) |
|---|---|---|---|---|---|
| 1. | 63.9 | 4.134 | 36.2 | 88.6 | 0.82 |
| 2. | 64.6 | 0.052 | 35.7 | 89.6 | 0.86 |
| 3. | 43.0 | 1.596 | 35.8 | 87.2 | 1.42 |
| 4. | 68.0 | 6.423 | 32.7 | 98.2 | 1.40 |
| 5. | 62.6 | 5.882 | 36.9 | 87.1 | 0.78 |
| 6. | 64.6 | 0.109 | 35.7 | 89.6 | 0.86 |
| 7. | 66.5 | 1.695 | 36.1 | 89.6 | 0.82 |
| 8. | 64.5 | 2.386 | 35.9 | 89.2 | 0.84 |
| 9. | 73.4 | 4.640 | 35.2 | 92.6 | 1.32 |
| 10. | 64.6 | 0.024 | 35.7 | 89.6 | 1.02 |
| 11. | 66.0 | 5.619 | 36.3 | 88.0 | 0.86 |
| 12. | 64.6 | 5.575 | 35.7 | 89.6 | 0.86 |
| 13. | 66.8 | 9.780 | 34.8 | 91.6 | 1.22 |
| 14. | 64.6 | 0.021 | 35.7 | 89.6 | 0.84 |
| 15. | 58.2 | 7.011 | 35.4 | 90.8 | 1.02 |
| 16. | 64.6 | 8.435 | 35.7 | 89.6 | 0.86 |

no visible trends in the random dependences. This is because of the small population, which is $NP = 2D$. Convergence is even slower if auto-adaptation is used. The algorithms tend towards stagnation from approximately the 40th generation. The second quaternion of tests (Fig. 3) worked with a bigger population, when $NP = 4D$. However, this population is still small; the algorithms exhibited the same behavior as in the first case.

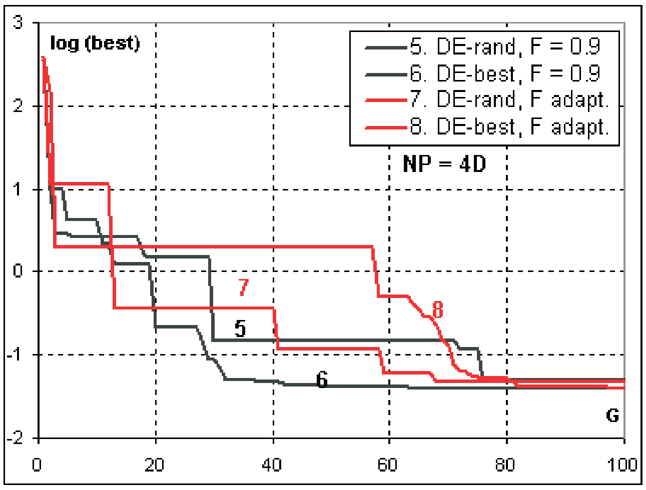

Fig. 3: Development of convergence in population $NP = 4D$

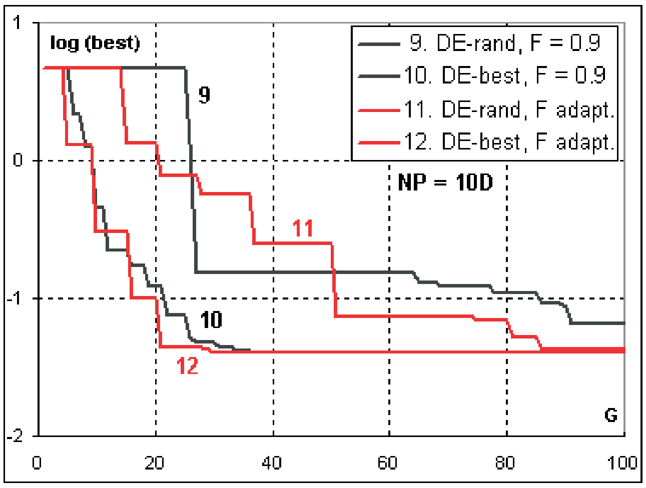

Fig. 4: Development of convergence in population $NP = 10D$

A better situation occurs when analysing algorithms working with a bigger population of about 10D. The dependences from Fig. 4 are more interesting; it is shown that variants *DE/best* exhibit better convergence. Variants *DE/best* established solutions faster and the final target function (*best*) is lower (courses 10 and 12 lay below courses 9 and 11). This trend can be observed in allmost all generations. There is a relatively inexpressive difference between test variants with and without adaptation of the *F* factor. Test variants with adaptation (courses 11 and 12) exhibit slightly faster convergence, but the differences are not outstanding.

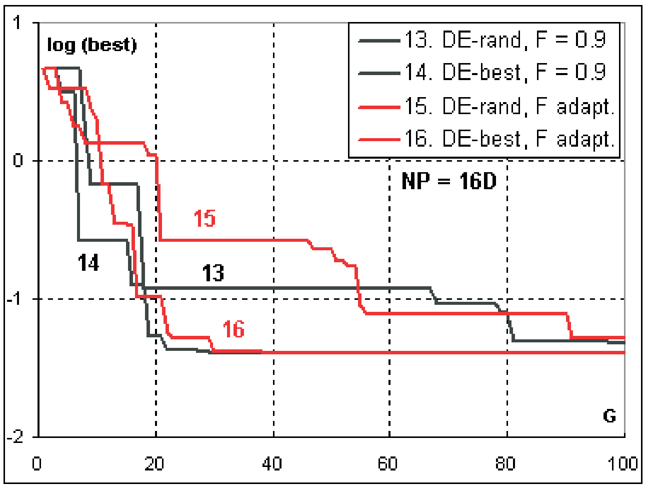

Fig. 5: Development of convergence in population $NP = 16D$

Almost the same results can be achieved when analyzing a population with $NP = 16D$ (Fig. 5). Variants *DE/best* again show better convergence than *DE/rand* and there is no expressive difference between the adapted and non-adapted variants. When comparing courses from Fig. 5, it should be pointed out that it has no sense to discuss generations younger than 10. At approximately that time, the solution will crystallize from random numbers. All initial populations were generated completely randomly.

## 5 Conclusion

We tested some variants of a *DE* algorithm for optimizing an electrical circuit. Attention was first focused on the difference between a *DE/rand* strategy and a *DE/best* strategy, and then on the influence of auto-adaptation. For this purpose a simple adaptation of control factor *F* was tested according to [1]. The tests showed a big influence of the choice of strategy, *DE/best* variants exhibit better results than *DE/rand* variants. The population must be big enough. The influence of auto-adaptation on convergence was not very expressive in this case, but variants with auto-adaptation of the *F* factor were somewhat faster. All tested variants find a solution in dedicated borders. The maximum divergence between the counted loss and the measured insertion loss was less than the 3 dB required by the CISPR standard.

## 6 References

[1] Tvrdík, J.: Evoluční algoritmy a adaptace jejich řídicích parametrů. *Automatizace*, Vol. **50** (2007), No. 7–8, p. 453–457.

[2] Zelinka, I.: *Umělá inteligence v problémech globální optimalizace*. Praha: BEN, 2002.

[3] ČSN CISPR 17 *Methods of Measurement of the Suppression Characteristics of Passive Radio Interference Filters and Suppression Components*.

[4] Internet: http://www.maplesoft.com/applications/view.aspx?SID=4680

Ing. Jiří Hájek
phone: +420 2 2435 2124
Fax: +420 2 2435 3949
e-mail: hajekj1@fel.cvut.cz

Department of Electrotechnology

Czech Technical University in Prague
Faculty of Electrical Engineering
Technická 2
166 27 Prague 6, Czech Republic