Paper 0123456789

©**Civil-Comp Press, 2005.**
Proceedings of the Eighth International Conference
on the Application of Artificial Intelligence
to Civil, Structural and Environmental Engineering,
B.H.V. Topping (Editor),
Civil-Comp Press, Stirling, Scotland.

Black-Box Function Optimization using Radial Basis Function Networks

A. Kučerová, M. Lepš and J. Skoček
Department of Structural Mechanics, Faculty of Civil Engineering
Czech Technical University in Prague, Czech Republic

Abstract

This paper is aimed at optimization of **black-box** functions. We assume that these functions are time demanding and therefore our goal is to minimize the number of evaluations of these functions. As one of the today's most promising algorithms, **the radial basis function network** (RBFN) is presented. Our particular implementation is based on the methodology presented in [12]. The novelty in our approach is the use of an evolutionary algorithm **GRADE** [7]. Also several scenarios of creating new points in the process of the approximation are presented. In comparison with the original approach [12], the number of needed evaluations of a test function is reduced approximately by a factor of two. To show the ability of the proposed methodology, the suite of twenty multi-modal functions is used along with one real-world problem of optimal control of structures undergoing large displacements.

Keywords: approximations, neural networks, radial basis function networks, global optimization, evolutionary algorithms, genetic algorithms, multi-modal problems.

1 Introduction

The ultimate goal of every optimization research is to develop a method capable of solving minimization problems for **black-box** functions. With this term we will understand unconstrained, real and often multi-modal functions without any knowledge of their derivatives or continuity. This is the case of many engineering problems usually connected with some form of a finite element analysis, which can cause non-linearity and discontinuity of a solved problem. We also assume that the optimized function is time demanding and therefore our goal is to minimize the number of evaluations of this function.

As one of the today's most promising algorithms, the **radial basis function net-**

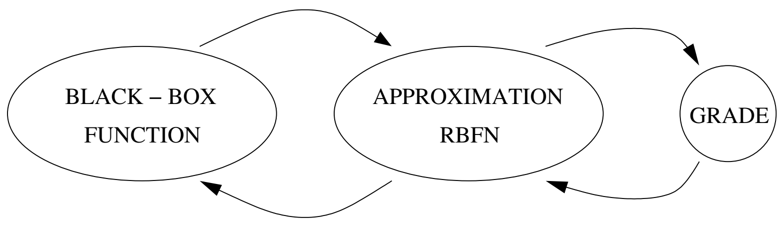

Figure 1: A scheme of the proposed algorithm

work (RBFN) is presented. This method comes from the domain of a general approximation, usually called Response Surface methods [9], Diffuse Approximations [7] or Surrogate models [8]. Methodology is based on an analogy with artificial neural networks, but differs in several points: the neural net is created only with one layer of neurons, it has a specific type of a transfer function and the training of this net leads to the solution of a linear system of equations.

Our particular implementation is based on the specifications presented in [12]. Particularly, the RBFN interpolates a given black-box function by the sum of basis functions values and neural weights for each neuron. The value of a basis function is influenced by its "distance" from the neuron's center to the vector of variables, for which we want to know the approximate function value. The neural weights are then derived from the condition of equality between the values of black-box function and its RBFN approximation.

The novelty in our approach is the use of the evolutionary algorithm GRADE [7] to find the global maximum of the RBFN approximation. Moreover, several scenarios of creating new points in the process of the approximation are presented.

The rest of the paper is organized as follows: Section 2 describes the two basic methods creating the core of the proposed procedure, namely the evolutionary algorithm GRADE and a particular implementation of a RBFN neural network. An attention is paid especially to the process of training RBNF as well as to the process of adding new points. Section 3 tries to show the ability of the proposed method to solve a suite of twenty multi-modal functions (already presented e.g. in [5]) along with one real-world problem of optimal control of structures undergoing large displacements [7]. The final Section 4 sums up several interesting remarks.

2 Description of used stochastic methods

The algorithm used herein to solve a **black-box** function is based on an efficient combination of an artificial neural network, namely **the radial basis function network** (RBFN) [12, 8], and an evolutionary algorithm GRADE [7]. The basic principle of this methodology is the interpolation of an objective function by the neural network and a search for the global optimum using the evolutionary algorithm. The simplified scheme of the optimization process is shown in Figure 1.

The main advantage of the proposed methodology is a fact that the evolutionary algorithm GRADE operates directly on an approximation of an expensive black-box function. This black-box function is evaluated only during the process of adding new points to the RBFN neural network. Therefore the task of adding new points into the RBFN is crucial for optimization as will be discussed later.

2.1 Evolutionary algorithm GRADE

Evolutionary algorithms belong to the most popular optimization methods nowadays. They follow up an analogy of processes that run in nature within the evolution of living organisms over a period of many millions of years. Unlike the traditional gradient optimization methods, evolutionary algorithms operate on so called population which is a set of possible solutions, applying the “genetic” operators, such as cross-over, mutation and selection. The principles of evolutionary algorithm was first proposed by Holland [4]. Ever since, the evolutionary algorithms have reached wide application domain (e.g. see the books of Goldberg [3] and Michalewicz [11] for extensive review).

Evolutionary algorithms in the original form operate on population of so-called chromosomes. These are binary strings which represent possible solutions in a certain way. In engineering problems we are usually working with real variables. Adaptation of the evolutionary algorithm idea to this problem was made possible by Storn [14] by considering the chromosomes as vectors instead of binary strings and using differential operators which can affect the distance between the chromosomes. In this work we employ an improved version of this kind of algorithm referred to as algorithm **GRADE** (GRadient-based Atavistic Differential Algorithm).

In the tradition of evolutionary methods, the first step is to generate a starting generation of chromosomes by choosing the random values of all state variables. Subsequently we repeat until convergence the cycles containing: the creation of a new generation of chromosomes, by mutation or cross-over, and the evaluation and selection, which reduces the actual number of chromosomes to the initial number. For the sake of clarity, a sketchy scheme of the algorithm GRADE follows:

Scheme of the algorithm GRADE

- The first operation is the `MUTATION`, which creates a new solution $\mathbf{x}$ using the operation
$$\mathbf{x} = \mathbf{y} + k(\mathbf{y} - \mathbf{z}) ,$$
where $\mathbf{y}$ is a solution from actual population, $\mathbf{z}$ is a randomly created solution and k is a real number from given bounds. This operation is used to ensure the diversity within a population.

- The second operation is the `CROSS-OVER`, which creates a new solution $\mathbf{x}$ by computing the difference between two existing solutions $\mathbf{y}$ and $\mathbf{z}$, multiplied by coefficients c and s, and adding the result to the better one of the first two solutions:

$$\mathbf{x} = \max(\mathbf{y}, \mathbf{z}) + cs(\mathbf{y} - \mathbf{z}) \ ,$$

where c is a real random number and s changes the direction of the descent $(\mathbf{y} - \mathbf{z})$ in favor to a better solution from vectors $\mathbf{y}$ and $\mathbf{z}$.

- The next operation `EVALUATION` computes the objective function value for each new solution.[1]

- The last and the most important operator in each cycle is the `TOURNAMENT SELECTION`, where the worst individual from two randomly selected solutions is deleted. This operator is repeated until the number of solution is the same as at the beginning of the cycle.

As a result of the optimization process, the solution with the highest objective function value is kept. For a more elaborate presentation of the proposed procedure see [7].

2.2 Radial Basis Function Network

Artificial neural networks (NNs)[2] were developed to simulate the processes in a human brain but later on it was discovered that they can be effectively used for many problems like pattern recognition, different approximations and predictions, control of systems, etc. In this work, they will be used "only" as general approximation tools.

A neural network is created with several neurons (here called perceptrons) which are mutually interconnected. In this work we will deal with so called **feed-forward**, **layered** neural networks, i.e. neurons form sorted layers, each layer is connected with the previous and the next layer and the signal is processed directly from the inputs neurons to the output ones.

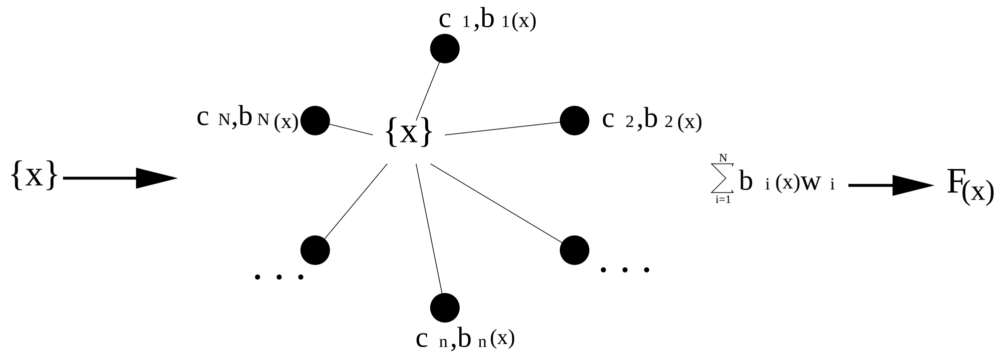

Figure 2: An approximation using RBFN

[1]Note that we assume maximization in this work.

[2]Hereafter we will use only the term *neural network* instead of *artificial neural network* for the sake of simplicity.

2.2.1 Approximation of a black-box function by RBFN

This type of a neural network is designed to simulate a black-box function $f(\mathbf{x})$ by its interpolation $F(\mathbf{x})$ given by the sum of basis functions multiplied by appropriate weights, see Figure 2. In other words,

$$f(\mathbf{x}) \approx F(\mathbf{x}) = \sum_{i=1}^{N} b_i(\mathbf{x}) w_i \ , \tag{1}$$

where $\mathbf{x}$ is a vector of unknowns, $b_i(\mathbf{x})$ is a basis function of the i-th neuron, w_i is a weight of the i-th neuron and N is the total number of neurons creating the net.

The basis function has the most often used “Gaussian” shape given by

$$b_i(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{c}_i\|^2/r} \ , \tag{2}$$

where $\mathbf{c}_i$ is a vector of coordinates of the center for the i-th basis function and r is a norm. Normalization ensures that basis functions will produce similar values for different scales in multidimensional spaces. The selection of the norm r is not crucial and therefore the most common form is used:

$$r = \frac{d_{max}}{\sqrt[dim]{dimN}} \ , \tag{3}$$

where d_{max} is the maximal distance within the domain, dim is the number of dimensions and N is the number of neurons.

2.2.2 Training of a neural net

The weights of individual neurons can be obtained by the process of “training”, see also Figure 3. Consider a set of training data

$$(\overline{\mathbf{x}_i}, \overline{y_i}) \ , \ i = 1, \ldots, p \ , \tag{4}$$

where $\overline{y_i}$ is a black-box function value in the $\overline{\mathbf{x}_i}$ point and p is a total number of records in the training set. For a usual RBFN the training set is identical with the basis functions centers, therefore we can write the training set also as

$$(\overline{\mathbf{c}_i}, \overline{y_i}) \ , \ i = 1, \ldots, N \ . \tag{5}$$

The weights w_i can be obtained from the equality between function values of a black-box function and its NN approximation in the function basis centers. Particularly,

$$f(\mathbf{c}_i) = F(\mathbf{c}_i) \tag{6}$$

and therefore

$$\min E = \min \sum_{i=1}^{N} [(\overline{y_i} - F(\mathbf{c}_i))^2 + \lambda_i w_i^2] \ , \tag{7}$$

where λ_i is used to regularize the system of equations (15) and in our computations it is set to $\lambda_i = 10^{-7}$. Inserting Equation (1) into (7) we get

$$\min E = \min \sum_{i=1}^{N} [(\overline{y_i} - \sum_{j=1}^{N} b_j(\mathbf{c}_i) w_j)^2 + \lambda_i w_i^2] \; . \tag{8}$$

To satisfy Equation (7) resp. (8), the following identity have to be fulfilled

$$\frac{\partial E}{\partial w_i} = 2 \sum_{i=1}^{N} [(\overline{y_i} - \sum_{j=1}^{N} b_j(\mathbf{c}_i) w_j)(b_i(\mathbf{c}_i)) + \lambda_i w_i] = 0 \; . \tag{9}$$

Define a matrix $[\mathsf{A}]_N$

$$[\mathsf{A}]_N = [\mathsf{B}]_N + [\Lambda]_N \; , \tag{10}$$

where $[\mathsf{B}]_N$ is a basis function matrix

$$[\mathsf{B}]_N = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]^T \; , \tag{11}$$

$$\mathbf{b}_i = [b_1(\mathbf{c}_i), b_2(\mathbf{c}_i), \ldots, b_N(\mathbf{c}_i)] \tag{12}$$

and a $[\Lambda]_N$ is a diagonal matrix with all λ_i non-zero members. Next, we can write a vector of outputs

$$\mathbf{w} = [w_1, w_2, \ldots, w_N] \tag{13}$$

and a vector of black-box function values

$$\overline{\mathbf{y}} = [\overline{y_1}, \overline{y_2}, \ldots, \overline{y_N}] \; . \tag{14}$$

Then, the Equation (9) becomes

$$[\mathsf{A}]_N \, \mathbf{w} = \overline{\mathbf{y}} \; . \tag{15}$$

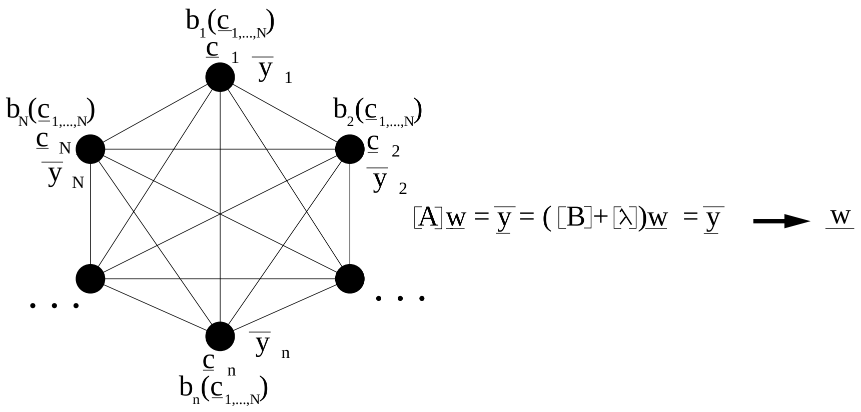
$b_1(\underline{c}_{1,\ldots,N})$
$\underline{c}_1$
$\overline{y}_1$
$b_2(\underline{c}_{1,\ldots,N})$
$\underline{c}_2$
$\overline{y}_2$
$b_N(\underline{c}_{1,\ldots,N})$
$\underline{c}_N$
$\overline{y}_N$
$[A]\underline{w} = \overline{\underline{y}} = ([B]+[\lambda])\underline{w} = \overline{\underline{y}} \longrightarrow \underline{w}$
. . .
. . .
$\overline{y}_n$
$\underline{c}_n$
$b_n(\underline{c}_{1,\ldots,N})$

Figure 3: Training of a neural net

By solving this system of *linear* equations we can obtain a vector of weights $\mathbf{w}$ for the given training set. Hence, from Equation (1), we can easily compute approximation $F(\mathbf{x}) \approx f(\mathbf{x})$ for any vector $\mathbf{x}$.

2.2.3 Adding new centers into RBFN

Recall the fact that we want to optimize the given black-box function. At this point, the RBFN approximation of the black-box function is created and hopefully, the optimum of the approximation and the original function will be located nearby. The above-mentioned evolutionary algorithm GRADE is used to find the optimum of the approximating function, i.e. the RBFN. It is obvious that especially in early stages, the RBFN is not able to interpolate correctly the given function and therefore new centers have to be added to minimize the difference between the RBFN and the black-box function. Keeping in mind that the number of black-box function calls is to be minimized, a step of adding new centers becomes very important.

To ensure a good approximation over the whole domain as well as maximum convergence speed of the proposed methodology, three scenarios of creating new centers are proposed. All of them add three new centers and always as a new point the best solution found by the algorithm GRADE from the previous cycle is added. Particularly, new centers are added as follows:

- the maximum found by GRADE and two centers created by a “hypercube method”,
- the maximum found by GRADE, one random center and one center selected by “variational method”,
- the maximum found by GRADE, one random center and one center created by “differential method”.

The **hypercube method** proposed in [12] uses a n-dimensional hypercube S created within the domain. Then, the first center is created inside this hypercube and the second outside. The inner center is added to speed up the convergence, the outer to improve global interpolation quality. The hypercube S is centered to the last found maximum and the length of a hypercube side is given by

$$l = l_0 \frac{1}{C_x + 1} , \tag{16}$$

where l_0 is a selected initial length and C_x is a number of already found optima in this hypercube during the whole optimization process.

The **variational method** creates one new center by

$$\mathbf{c}_{new} = \mathbf{c}_j + var \cdot (\mathbf{c}_j - \mathbf{c}_{j-1}) , \tag{17}$$

where $\mathbf{c}_j$ is a maximum found in the actual cycle, $\mathbf{c}_{j-1}$ is a maximum from the previous cycle and var is the variance computed from the j previous cycles.

The **differential method** adds only one point in the sense of the “numerical” steepest descent, i.e.

$$\mathbf{c}_{new} = \mathbf{c}_j + u(0,1)(\mathbf{c}_j - \mathbf{c}_{j-1}) , \tag{18}$$

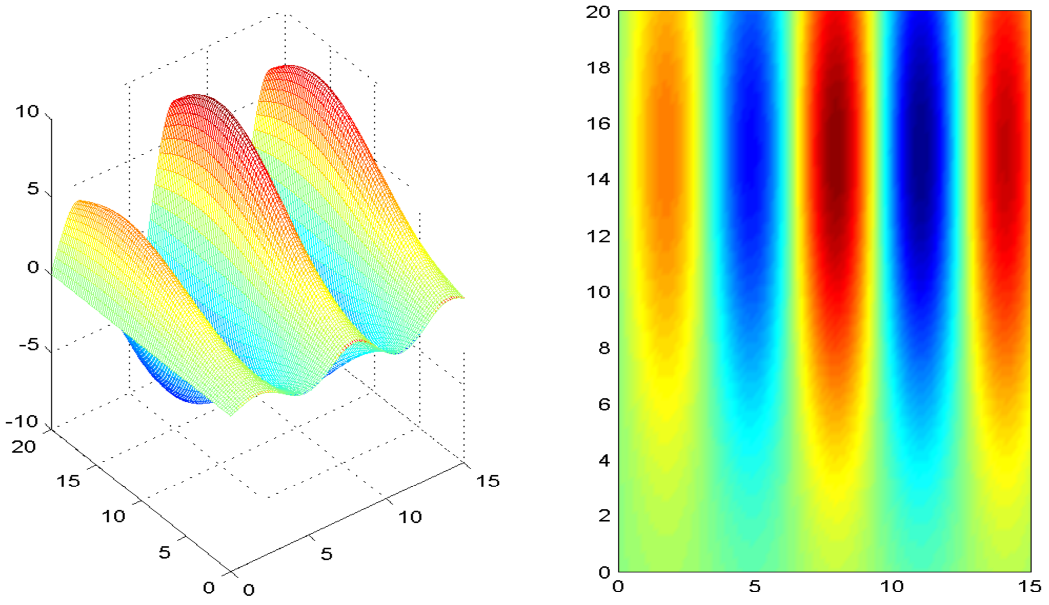

Figure 4: Function *ex1*

in the case that the function value of $\mathbf{c}_j$ is greater then $\mathbf{c}_{j-1}$ and

$$\mathbf{c}_{new} = \mathbf{c}_j + u(0,1)(\mathbf{c}_{j-1} - \mathbf{c}_j) \tag{19}$$

otherwise.[3]

To compare these three methods, the test function called *ex1* was examined, see Figure 4 or reference [12]. The function *ex1* is given by

$$f(\mathbf{x}) = f(x,y) = 10e^{-0.01(x-10)^2 - 0.01(y-15)^2} \sin(x) \; . \tag{20}$$

The domain is bounded to $\langle 0, 15 \rangle \times \langle 0, 20 \rangle$ and the maximum is $f(7.8960, 15.0000) = 9.5585$. If the difference between the approximation and the *ex1* function falls under 10^{-4}, the optimization process is stopped. Each of the three methods was run one hundred times and the statistics was stored. The results are shown in Table 1.

	Method used		
No. evaluations	Hypercube	Variational	Differential
Minimum	41	29	29
Maximum	207	218	68
Std. dev.	31.89	45.50	15.48
Average	126.46	44.90	**42.32**

Table 1: Comparison of individual methods

[3]Note that $u(a,b)$ denotes a continuous random variable uniformly distributed on an interval (a,b).

It is clearly visible that the differential method needs in average a minimum number of evaluations. Also the deviation in results for the differential method is the smallest among all three methods. The progress in comparison with referenced hypercube method is noteworthy. Hence all following computations are done using the differential version of our RBFN methodology. To conclude this section, the detailed description of individual steps of proposed procedure follows:

1. Create initial neurons,
2. Compute parameters of basis functions d_{max} and r,
3. Repeat next steps until stopping criteria are met
4. Compute black-box function values in new center points
5. Compute values of basis functions $b_i(\mathbf{c}_i)$ and output weights w_i,
6. Find a maximum using the evolutionary algorithm GRADE,
7. Add three new points using the differential method.

3 Solved examples

To show the abilities of the proposed methodology, the set of twenty multi-modal functions, firstly compiled in [1] and later used e.g. in [5], was used. The obtained results were compared to those presented in the above mentioned papers.

The second test example shows the applicability of the RBFN for practical tasks. The optimal control of a structure in the domain of large deformations [7] is conducted and results are compared to alternative optimization methods.

3.1 The set of twenty multi-modal functions

The used set contains the suite of twenty well-known multi-modal functions known from optimization area. The complete list of these functions can be found in Appendix A. The proposed algorithm is compared with following evolutionary methods:

- `SBGA` Standard Binary Genetic Algorithm [1],
- `EBGA` Extended Binary Genetic Algorithm [1],
- `DE` Differential Evolution [15], [6],
- `SADE` Simplified Atavistic Genetic Algorithm [6], [5],
- `GRADE` GRadient-based Atavistic Differential Algorithm [7] (for details, see also Section 2.1).

The statistics over one hundred runs was computed to tackle random circumstances. The optimization process was stopped after 300 cycles (equals to 900 evaluations) or if the maximum was found with the given precision. Otherwise, the optimization run was marked as unsuccessful. The statistics from one hundred runs follows:

		No. successful	**No. evaluations**		
Function	**Dim.**	**runs**	**Minimum**	**Maximum**	**Average**
F1	1	100	12	36	19.44
F3	1	40	12	144	69.15
Branin	2	100	21	111	46.20
Camelback	2	100	9	87	54.21
Goldprice	2	1	321	321	321
PShubert1	2	1	36	36	36
PShubert2	2	2	303	348	325.5
Quartic	2	100	18	117	78.48
Shubert	2	20	264	846	481.62
Hartman1	3	97	15	321	83.20
Shekel1	4	0	-	-	-
Shekel2	4	0	-	-	-
Shekel3	4	0	-	-	-
Hartman2	6	39	162	351	252
Hosc45	10	100	465	852	625.98
Brown1	20	0	-	-	-
Brown3	20	0	-	-	-
F5n	20	0	-	-	-
F10n	20	0	-	-	-
F15n	20	0	-	-	-

Table 2: Results for the function set

A comparison among evolutionary algorithms is presented in Table 3. In Table 3, **Fc** stands for an average number of function calls and **Suc** for a number of successful runs out of one hundred.

From the Table 3 it is obvious that the proposed combination of RBFN and evolutionary algorithm cannot be aimed at optimization of multi-modal functions. On the other hand, the ability of the proposed algorithm to solve problems with a smaller number of local minima is remarkable.

3.2 Optimal control of structures under large deformations

The aim of this section is to show the possibilities of the proposed procedure in solving practical problems. As an example, the optimal control of a "T"-shaped beam is solved, see Figure 5. The goal is to find the values of a loading force and a bending moment such that the final shape is obtained. The objective function is defined as

Table 3: Comparison of evolutionary algorithms

		SBGA		EBGA		DE		SADE		GRADE		RBFN
Function	**Dim.**	**Suc**	**Fc**	**Suc**	**Fc**	**Suc**	**Fc**	**Suc**	**Fc**	**Suc**	**Fc**	**Suc**	**Fc**
F1	1	100	5566	100	784	100	52	100	72	100	52	100	19
F3	1	100	5347	100	744	100	98	100	88	100	88	40	69
Branin	2	81	8125	100	2040	100	506	100	478	100	383	100	46
Camelback	2	98	1346	100	1316	100	244	100	273	100	200	100	54
Goldprice	2	59	8125	100	4632	100	350	100	452	100	334	1	321
PShubert1	2	63	7192	100	8853	83	1342	100	2738	100	5211	1	36
PShubert2	2	59	7303	100	4116	90	908	100	1033	100	1932	2	326
Quartic	2	83	8181	100	3168	97	313	100	425	100	307	100	78
Shubert	2	93	6976	100	2364	94	10,098	100	585	100	632	20	482
Hartman1	3	94	1993	100	1680	100	284	100	464	100	305	97	83
Shekel1	4	1	7495	97	36,388	72	1968	99	61,243	100	51,461	0	-
Shekel2	4	0	-	98	36,774	91	1851	100	17,078	100	15,380	0	-
Shekel3	4	0	-	100	36,772	89	1752	99	11,960	100	5548	0	-
Hartman2	6	23	19,452	92	53,792	16	4241	67	2297	59	199,502	39	252
Hosc45	10	0	-	2	126,139	100	1174	100	6438	100	2153	100	626
Brown1	20	0	-	0	-	100	65,346	95	163,919	100	182,234	0	-
Brown3	20	5	8410	5	106,859	100	41,760	100	43,426	100	35,480	0	-
F5n	20	0	-	100	99,945	96	38,045	66	17,785	100	7264	0	-
F10n	20	0	-	49	113,929	90	71,631	47	110,593	100	90,794	0	-
F15n	20	0	-	100	102,413	100	44,248	93	28,223	100	23,102	0	-

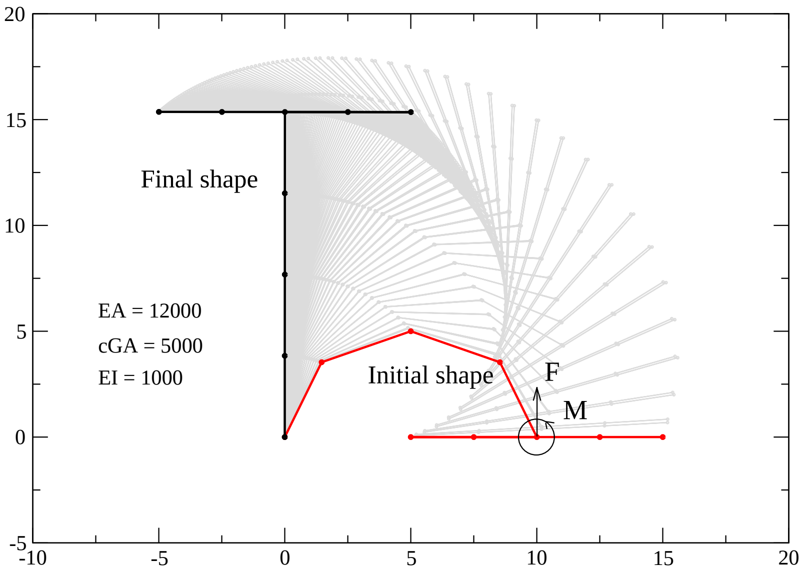

Figure 5: An example of a T-shaped beam

Algorithm	Average number of function calls
SADE	648.8
GRADE	512.4
RBFN	104.2

Table 4: Comparison of evolutionary methods in solving optimal control problem

a least square between the final and the desired displacements. The structural analysis part takes into account large deformations and therefore the optimization problem is highly non-linear. For more detail on this problem, see again [7]. The results in the terms of average number of function calls are presented in Table 4.

4 Conclusions

It is evident from Table 3 that once the proposed methodology correctly approximates the black-box function (the cases with 100 % successful runs) the drastic reduction of function calls is observed. In the other cases, the procedure cannot be advised for solving highly multi-modal functions without any tool for management of local optima, see e.g. works [10] or [5]. Nevertheless, as indicated by the results stored in Table 4, it can be expected that common engineering optimization tasks (such as optimal design and/or control of structures) will lead to problems with only a limited number of local optima. This increases the potential applicability of the proposed approach. However, besides the problems with convergence, several difficulties (such as numerical stability of the linear system of equations, a fully populated matrix to be stored etc.) during the growth of neural centers were noted. In the near future, this will be solved by clustering techniques or by orthogonalization method, see e.g. [2].

Acknowledgements The financial support of this work by research project of the Czech Ministry of Education, MSM 6840770003, and by Grant Agency of the Czech Republic with grant GACR 103/05/H506 is gratefully acknowledged.

References

[1] J. Andre, P. Siarry, and T. Dognon, *An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization*, Advances in Engineering Software **32** (2000), no. 1, 49–60.

[2] John A. Bullinaria, *Introduction to neural networks - course material and useful links*, 2005, `http://www.cs.bham.ac.uk/~jxb/inn.html`.

[3] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[5] O. Hrstka and A. Kučerová, *Improvements of real coded genetic algorithms based on differential operators preventing the premature convergence*, Advances in Engineering Software **35** (2004), no. 3–4, 237–246.

[6] O. Hrstka, A. Kučerová, M. Lepš, and J. Zeman, *A competitive comparison of different types of evolutionary algorithms*, Computers & Structures **81** (2003), no. 18–19, 1979–1990.

[7] A. Ibrahimbegović, C. Knopf-Lenoir, A. Kučerová, and P. Villon, *Optimal design and optimal control of elastic structures undergoing finite rotations*, International Journal for Numerical Methods in Engineering **61** (2004), no. 14, 2428–2460.

[8] Marios K. Karakasis and Kyriakos C. Giannakoglou, *On the use of surrogate evaluation models in multi-objective evolutionary algorithms*, in Neittaanmäki et al. [13].

[9] Jongsoo Lee and Prabhat Hajela, *Application of classifier systems in improving response surface based approximations for design optimization*, Computers & Structures **79** (2001), 333–344.

[10] Samir W. Mahfoud, *Niching methods for genetic algorithms*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3^{rd} ed., Springer-Verlag, 1999.

[12] H. Nakayama, K. Inoue, and Y. Yoshimori, *Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges*, in Neittaanmäki et al. [13].

[13] P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, P. Périaux, and D. Knörzer (eds.), *European congress on computational methods in applied sciences and engineering (ECCOMAS 2004)*, Jyväskylä, 2004.

[14] R. Storn, *On the usage of differential evolution for function optimization*, NAPHIS 1996, Berkeley, 1996, pp. 519–523.

[15] R. Storn and K. Price, *Differential Evolution : A simple and efficient adaptive*

scheme for global optimization over continuous spaces, Tech. Report TR-95-012, University of Berkeley, 1995.

A List of test functions

- F1:

$$f(x) = 2(x-0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125 \; , \tag{21}$$

where

$$0 \leq x \leq 1$$

- F3:

$$f(x) = -\sum_{j=1}^{5} [j \sin[(j+1)x + j]] \; , \tag{22}$$

where

$$-10 \leq x \leq 10$$

- Branin:

$$f(x,y) = a(y - bx^2 + cx - d)^2 + h(1-f)\cos x + h \; , \tag{23}$$

where

$$a = 1, b = 5.1/4\pi^2, c = 5/\pi, d = 6 \; ,$$

$$h = 10, f = 1/8\pi, -5 \leq x \leq 10, 0 \leq y \leq 15$$

- Camelback:

$$f(x,y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right) x^2 + xy + (-4 + 4y^2)y^2 \; , \tag{24}$$

where

$$-3 \leq x \leq 3, -2 \leq y \leq 2$$

- Goldprice:

$$\begin{aligned} f(x,y) = \; & [\; 1 + (x+y+1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)] \cdot \\ & [\; 30 + (2x-3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)] \; , \end{aligned} \tag{25}$$

where

$$-2 \leq x \leq 2, -2 \leq y \leq 2$$

- PShubert 1 a 2:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x+i]\right\} \cdot \left\{\sum_{i=1}^{5} i\cos[(i+1)y+i]\right\} + \beta[(x-1.42513)^2+(y+0.80032)^2]\,, \tag{26}$$

where

$$-10 \le x \le 10, -10 \le y \le 10,$$

pro PShubert1: $\beta = 0.5$
pro PShubert2: $\beta = 1.0$

- Quartic:

$$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}\,, \tag{27}$$

where

$$-10 \le x \le 10, -10 \le y \le 10$$

- Shubert:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x+i]\right\} \cdot \left\{\sum_{i=1}^{5} i\cos[(i+1)y+i]\right\}\,, \tag{28}$$

where

$$-10 \le x \le 10, -10 \le y \le 10$$

- Hartman 1:

$$f(x_1,x_2,x_3) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{3} a_{ij}(x_i-p_{ij})^2}\,, \tag{29}$$

where

$$0 \le x_i \le 1, i = 1, ..., 3$$

$$x = (x1, ..., x3), p_i = (p_{i1}, ..., p_{i3}), a_i = (a_{i1}, ..., a_{i3})$$

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.36890	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.46990	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.10910	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.03815	0.5743	0.8828

- Shekel 1,2 a 3:

$$f(x,y) = -\sum_{i=1}^{m} \frac{1}{(x_i - a_i)^T(x_i - a_i) + c_i} \,, \qquad (30)$$

where

$$0 \le x_j \le 10,$$

pro Shekel1: $m = 5$,
pro Shekel2: $m = 7$,
pro Shekel3: $m = 10$

$$x = \{x_1, x_2, x_3, x_4\}^T, a_i = \{a_{i1}, a_{i2}, a_{i3}, a_{i4}\}^T$$

i	a_{ij}				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.6
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

- Hartman 2:

$$f(x_1, ..., x_6) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{6} a_n (x_i - p_{ij})^2} \,, \qquad (31)$$

where

$$0 \le x_j \le 1, j = 1, ..., 6$$

$$x = (x_1, ..., x_6), p_i = (p_{i1}, ..., pPi6), a_i = (a_{i1}, ..., a_{i6})$$

i	a_{ij}						c_i
1	10.00	3.00	17.00	3.50	1.70	8.00	1.0
2	0.05	10.00	17.00	0.10	8.00	14.00	1.2
3	3.00	3.50	1.70	10.00	17.00	8.00	3.0
4	17.00	8.00	0.05	10.00	0.01	14.00	3.2

i	a_{ij}					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

- Hosc 45:

$$f(x) = 2 - \frac{1}{n!}\prod_{i=1}^{n} x_i \ , \tag{32}$$

where

$$x = (x_1, ..., x_n), 0 \leq x_i \leq i, n = 10$$

- Brown 1:

$$\begin{aligned} f(x) = \quad & \left[\sum_{j\in J}(x_i - 3)\right]^2 + \\ & \sum_{j\in J}[10^{-3}(x_i-3)^2 - (x_i - x_{i+1}) + e^{20(x_i - x_{i+1}}] \ , \end{aligned} \tag{33}$$

where

$$J = \{1, 3, ..., 19\}, -1 \leq x_i \leq 4, 1 \leq i \leq 20 \ , x = (x_1, ..., x_{20})^T$$

- Brown 3:

$$f(x) = \sum_{i=1}^{19}[(x_i^2)^{(x_{i+1}^2+1} + (x_{i+1}^2)^{(x_i^2+1)}] \ , \tag{34}$$

where

$$x = \{x_1, ..., x_{20}\}^T, -1 \leq x_i \leq 4, 1 \leq i \leq 20$$

- F5n:

$$\begin{aligned} & f(x) = (\pi/20)\cdot \\ & \left\{10\sin^2(\pi y_1) + \sum_{i=1}^{19}[(y_i-1)^2\cdot(1+10\sin^2(\pi y_i+1))] + (y_{20}-1)^2\right\} \ , \end{aligned} \tag{35}$$

where

$$x = \{x_1, ..., x_{20}\}^T, -10 \leq x_i \leq 10, y_i = 1 + 0.25(x_i - 1)$$

- F10n:

$$\begin{aligned} & f(x) = (\pi/20)\cdot \\ & \left\{10\sin^2(\pi x_1) + \sum_{i=1}^{19}[(x_i-1)^2\cdot(1+10\sin^2(\pi x_{i+1}))] + (x_{20}-1)^2\right\} \ , \end{aligned} \tag{36}$$

where

$$x = \{x_1, ..., x_{20}\}^T, -10 \leq x_i \leq 10$$

- F15n:

$$\begin{aligned} & f(x) = (1/10) \cdot \\ & \{\sin^2(3\pi x_1) + \sum_{i=1}^{19}[(x_i - 1)^2(1 + sin^2(3\pi x_{i+1}))] + \\ & (1/10)(x_{20} - 1)^2[1 + \sin^2(2\pi x_{20})]\}\ , \end{aligned} \tag{37}$$

  where

$$x = \{x_1, ..., x_{20}\}^T, -10 \leq x_i \leq 10$$