European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2004
P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau (eds.)
H. W. Engl and F. Tröltzsch (assoc. eds.)
Jyväskylä, 24–28 July 2004

MATERIAL PARAMETERS IDENTIFICATION FOR MICROPLANE MODEL BY GENETIC ALGORITHM

Anna Kučerová, Matěj Lepš, and Zdeněk Bittnar

Department of Structural Mechanics, Faculty of Civil Engineering,
Czech Technical University in Prague, Thákurova 7, 166 29 Prague 6, Czech Republic
e-mail: anicka@cml.fsv.cvut.cz, web page: http://cml.fsv.cvut.cz/∼anicka
e-mail: leps@cml.fsv.cvut.cz, web page: http://cml.fsv.cvut.cz/∼leps

Key words: Genetic Algorithms, Microplane Model, Parameter Estimation, Optimization, Parallel Computing.

Abstract. *In this contribution, an application of a genetic algorithm-based optimizer to material model parameters identification is discussed. Particularly, an optimization algorithm based on an efficient combination of Genetic Algorithm ideas with the principles of the Differential Evolution method is presented as a convenient tool for such type of problems. The current application of this method to the microplane constitutive model [1] reveals the limitations of the formulated identification problem and offers possible means for its regularization. To bring this method closer to real use, the parallel implementation is proposed and implemented.*

1 INTRODUCTION

The growth of computer availability as well as their performance enables new developments within many research areas especially during last few decades. In particular, within civil and material engineering this fact allows efficient optimization of various real-world problems. In these applications, however, the function to be optimized is often non-smooth or it can be even discontinuous or discrete, which presents critical difficulties to traditional optimization methods. In contrary, Genetic algorithms (GAs), as one of the most prominent representatives of global optimization methods, generally need only the values of the optimized function and eventually the bounds for each parameter. This versatility is quite naturally compensated by a large number of objective function calls that inevitably results in an increased demand on computational resources, which is, however, nowadays met for common personal computers.

One of the main impulses for the development of the genetic algorithms-based optimizers at our department was search for the method that can be used for neural network training with an emphasis given to the subsequent application of the neural network to the prediction of material parameters of microplane constitutive model [5]. Note that the training process of a neural network is actually a highly-dimensional optimization problem with optimized variables representing the weights in the artificial neural network. The SADE algorithm, described in the following section, is the result of this particular study. Although the proposed algorithm succeeded in neural network training and outperformed traditional techniques [5], resulting predictions of the neural network show a significant error. To obtain more reliable results, the applicability of the global optimization algorithm for the parameter identification problem is examined in this work. Despite the minimization of computational requirements by using smaller problem, the parallelization of the program was necessary.

The rest of the paper is organized as follows. The next two sections are devoted to description of the optimization strategy SADE. The microplane material model is introduced in the fourth section. The parallelization strategy together with obtained results are discussed at the end of this paper.

2 SADE ALGORITHM

For the purpose of evaluating the influence of a number of variables on algorithm behavior, a simple but representative case study with varying number of variables was elaborated [6]. First, this study was executed for the Differential Evolution algorithm [11, 12] and, quite surprisingly, very rapid increase of the number of function calls was observed. Therefore, several modification of the Differential Evolution were proposed [6], which eventually led to the SADE[1] algorithm with the linear increase in a computation cost. The resulting algorithm has the following C++ programming language form:

[1] Simplified Atavistic Differential Evolution

```
void SADE ( void )
{
   FIRST_GENERATION ();
   while ( to_continue )
   {
      MUTATE ();
      LOCAL_MUTATE ();
      CROSS ();
      EVALUATE_GENERATION ();
      SELECT();
   }
}
```

3 DESCRIPTION OF FUNCTIONS

In this section, each of the steps of the introduced algorithm is described in more detail. For further information, we refer an interested reader to the work [6] or the Internet page [8]. Note that in the following text, an attention is paid to the constants that the algorithm is working with since they have a cardinal impact on the algorithm performance. It should be emphasized that their optimal setting is unfortunately problem-dependent and presents additional non-trivial task that must be dealt with.

3.1 FIRST_GENERATION

In this function, called at the beginning of the process, the population of individuals, or vectors, whose coordinates are random numbers within given bounds, is generated. There is only a single constant in this function that defines how large population of the individuals is to be generated. Based on some test problems, it was concluded that it is convenient to select this number as linearly dependent on the number of variables of a solved objective function. In particular, to get the size of the population, the parameter `pool_rate` is used to multiply the number of optimized variables. For majority of solved functions the algorithm was successful with `pool_rate = 10`; occasionally, it was necessary to take its value around `30`. An essential property of this parameter is the ability to slow down the convergence process, which prevents the algorithm to fall into a local extreme.

3.2 MUTATE

Within the function `MUTATE`, a certain number of new individuals is created by mutation of the current population. The procedure works as follows: an individual A is randomly chosen from the population, further a new individual B is created with all coordinates as random numbers within given bounds. A new individual, which is to be added to the population, is created by translation of the vector A by a random fraction of the distance AB in the direction of the vector B. For such a form of the mutation operator,

a new constant describing the number of individuals to be created by this operator is introduced. This constant, called `radiation`, should range from 0 to 30% of the number of individuals at the beginning of the cycle. Often it is chosen to be equal to 10%. The bigger the constant is, the larger scatter of individuals in the population occurs and the convergence slows down. Attention must be paid to the size of this parameter, because for some values the algorithm failed to converge at all.

3.3 LOCAL_MUTATE

The goal of this operator is to increase algorithm's performance for problems where a solution with a high precision is sought. It again introduces a certain number of new individuals in the population by translating all coordinates of a randomly chosen vector from the population by a small random distance. The translation is chosen for each coordinate separately as the fraction of the range for the given variable with the fraction randomly chosen from the interval -0.0025 to 0.0025.

As in the case of the previously described operator MUTATE, it is again necessary to define the number of individuals to be created by local mutation. The constant called `local_radiation` defines this number as the percentage of the total number of individuals in the population at the beginning of the cycle. The constant is set within the range from 0% to 30%, the typical value is 10%.

3.4 CROSS

This operator creates new individuals so that, at the end of one generation, their total number is twice the number of individuals at the beginning of the cycle. The new individual is created on the principle of differential crossing. Three individuals A, B, C are randomly chosen from the population. New individual D is created according to the following relation,

$$D = A + \texttt{cross\_rate} \cdot (B - C). \tag{1}$$

The parameter `cross_rate` has the largest influence on the convergence of the algorithm. Its value is usually taken from the range 0.1 to 0.5. The bigger is its value, the slower is convergence of the algorithm.

3.5 EVALUATE_GENERATION

The function `EVALUATE_GENERATION` evaluates all new individuals. Firstly, it is necessary to define the manner how to treat the individuals (vectors), which fall outside the user-defined bounds. If the optimized function is defined beyond these margins, it is sometimes advantageous not to penalize these individuals. Such behavior enables to find new solutions if the margins are not well-known in advance. In the opposite case, where the solution outside the margins is not feasible, a suitable form of penalization should be

used. The SADE algorithm uses in this case the so-called 'boundary return', which means that the coordinate beyond the margins is replaced with the nearest boundary value.

3.6 SELECT

This operator reduces the number of individuals to one half, i.e. the same number as it was at the very beginning of the cycle. This is done on the basis of natural selection; in particular, the inverse tournament selection is performed: from two randomly selected individuals the worse is disqualified from the population. This selection is random and no additional tuning parameter is needed.

3.7 Parameters settings

As a whole, only four constants are defined in the SADE algorithm. For the reported optimization problem, these parameters were set as follows,

```
pool_rate = 10
radiation = 5%
local_radiation = 5%
cross_rate = 0.1
```

4 MICROPLANE MODEL PARAMETERS IDENTIFICATION

This section deals with the above presented SADE method applied to the estimation of parameters of a constitutive model for concrete called *microplane model*. Concrete is a heterogeneous man-made material and therefore the simulation of its behavior encounters serious difficulties, both theoretical and numerical. The microplane model [1, 2, 7] is a fully three-dimensional material law that includes tensional and compressive softening, damage of the material, different combinations of loading, unloading and cyclic loading. It can describe the development of anisotropy within the material. The major disadvantage of this model, however, is an enormous computational cost associated with structural analysis and a large number of phenomenological material parameters. Therefore, a reliable procedure for parameters identification is on demand. In particular, a certain type of concrete is described by eight parameters: Young's modulus E, Poisson's ratio ν, and other six parameters (k_1, k_2, k_3, k_4, c_3, c_{20}), which do not have a simple physical interpretation, and therefore it is difficult to determine their values from experiments.

The common practice for an experimenter is to employ a trial and error method to tune stress-strain diagrams by varying the model parameters [2, 10]. This is not trivial task because of highly non-linear behavior, however, several limits can be found in the literature for these parameters. In the current implementation, the appropriate bounds were set to values shown in the Table 1.

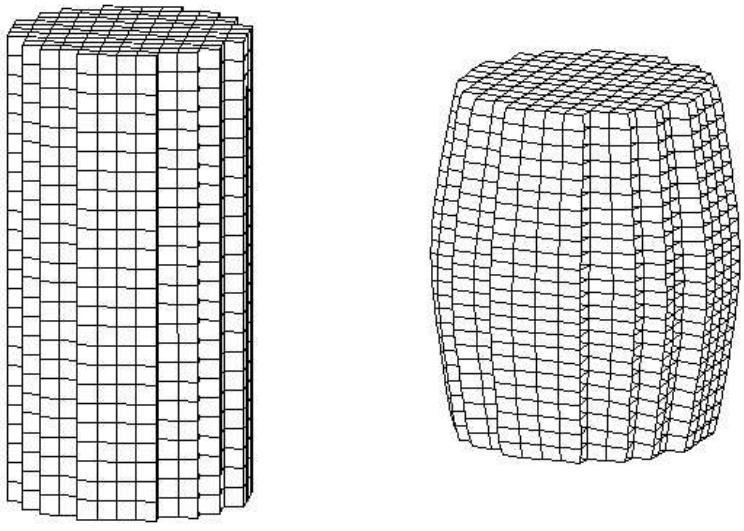
To define the problem more formally, the optimization goal is to find microplane parameters from the stress-strain diagram of a test specimen in a uniaxial compression, see Figure 1. The objective function is then the least square error function, which contains

$$
\begin{array}{rcl}
E & \in & \langle 15.0, 50.0 \rangle \ \text{GPa} \\
\nu & \in & \langle 0.1, 0.4 \rangle \\
k_1 & \in & \langle 0.0, 5.0 \rangle \\
k_2 & \in & \langle 100.0, 1000.0 \rangle \\
k_3 & \in & \langle 5.0, 15.0 \rangle \\
k_4 & \in & \langle 30.0, 200.0 \rangle \\
c_3 & \in & \langle 3.0, 5.0 \rangle \\
c_{20} & \in & \langle 0.2, 5.0 \rangle
\end{array}
$$

Table 1: Boundaries for the microplane model parameters.

the difference between values of a known stress-strain curve (from an experiment or from structural analysis) and values from the microplane model simulation.

The rather severe disadvantage of a microplane model is an extreme demand of computational time. As it was shown in [10], the above presented example of a uniaxial test (Fig. 1) consumed more than 23 hours on a single processor PC with the Pentium II Xeon 400 MHz processor and 512 MB RAM. Therefore, a single finite element is used instead of the whole specimen. It was demonstrated in [10] that this simplified model is not so unrealistic as it may appear on the first sight; the differences in fitted parameters found by these two approaches were not significant. Even though, the total computational time is not negligible. Results for one element non-linear analysis on different types of single processor PCs are presented in Table 2. The computations were executed one hundred times with randomly chosen material parameters and minimum, average and maximum values are shown. Note the big differences between minimum and maximum times, which



(a) (b)

Figure 1: The computational model of a compression test a) at the start and b) at the end of loading.

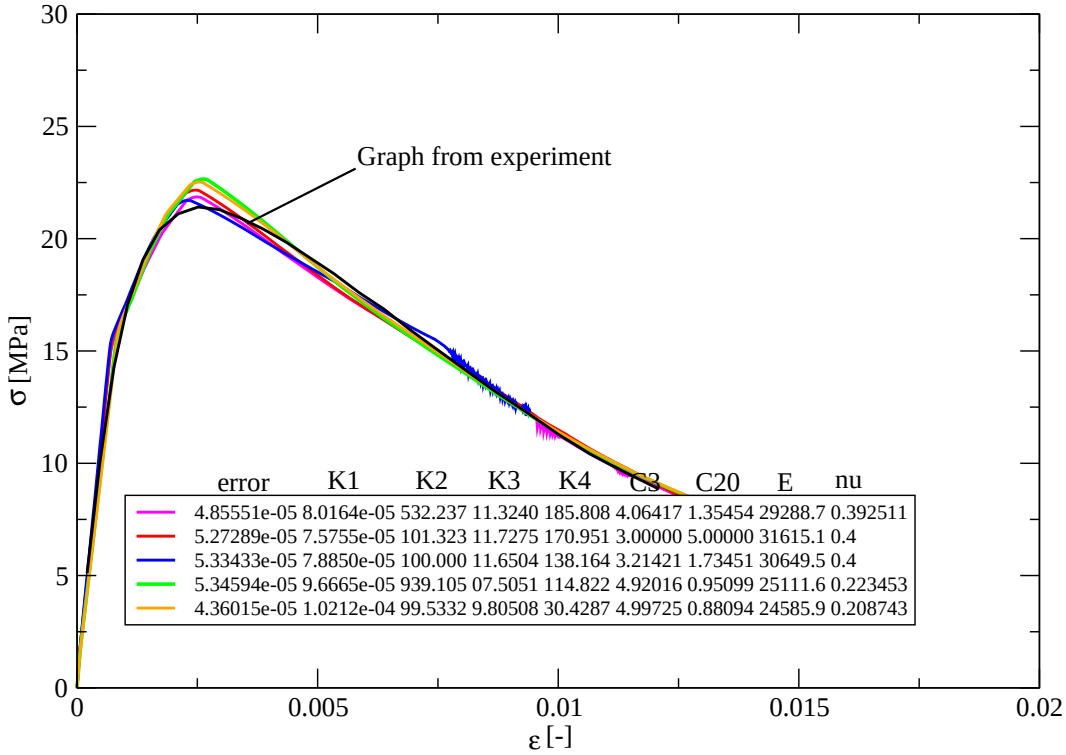


Figure 2: Five different results and an original stress-strain curve.

result from the non-linear response of the structure.

The Figure 2 shows an example of the "real" stress-strain curve from an experiment, and its five, locally optimal, approximations with a microplane model. These results were obtained after 3,000 evaluations of the optimization method.

5 PARALLELIZATION SCHEME

Implementing the optimization method into the parallel environment, the independence of individual solutions within one generation of genetic algorithm, often called "implicit parallelism" [9], is used: the search toward an optimum solution is governed by the population of possible solutions and their recombination. When new individuals are created, there is a need for an estimation how good new solutions are. In this phase, the division for parallel computing is made so that the root computer runs the optimization

Processor	Min. [s]	Avg. [s]	Max. [s]
Intel Pentium III 450 MHz	47.96	89.76	213.32
Intel Pentium III 1000 MHz	22.45	47.95	97.73
Intel Xeon 1700 MHz	16.80	38.63	87.81

Table 2: Obtained times of one computation on different processors.

algorithm together with a small portion of the whole population while the slave processors solve the remaining solutions. In [3], this model is called *Global parallelization*. More specifically, the *synchronous* version, where the master process waits for responses from slaves, is employed. The advantages of this model are independence on the hardware platform, i.e. can be run on shared memory systems as well as on the distributed memory ones, and the fact that the global parallel model produces the same behavior as his serial ancestor, mainly the same number of internal (tuning) parameters.

The chosen parallel computing scheme ensures constant distribution of the work among the processors provided that the time spent on evaluation of two solutions does not differ. Although this condition is not strictly met for the current problem (see Table 2), it appears that the sufficiently high number of solutions assigned to each processor eliminates this disadvantage.

To be sure that the parallel algorithm is well-designed with respect to the number of available processors, the optimum amount of processors is checked. It can be simply estimated by following relation [4]

$$P^* = \sqrt{\frac{nT_f}{T_c}} \doteq 440 \quad \text{processors} \tag{2}$$

where

P^*	is the optimal number of processors,
n	is the number of solutions in population, in our case 80,
T_f	is time for one evaluation of a function, set to 47.95 s, see Table 2,
T_c	is latency time - hardware dependent variable, which is spent on creating communication between two processors, in our case equal to 20 ms.

It is clear that in this particular case, the linear speedup can be expected even for substantially higher number of processor than it is available at the author's research department. Therefore, the obtained near-linear speedup is nothing surprising (see Fig. 3).

Resulting times were obtained by running the presented optimization problem on a PC cluster, installed at the Department of Structural Mechanics, Faculty of Civil Engineering, CTU in Prague. The PC cluster consists of ten two-processor DELL workstations. Parameters of the three used computers (the fastest ones) are described in Table 3. The workstations are connected by Fast Ethernet 100 Mb network using 3Com Superstack II switch. Note that this cluster represents a heterogeneous parallel computing platform.

The total times for the present problem are summarized in Fig. 4. The results were obtained for 880 evaluations (10 generations per 80 solutions, the first generation needs twice more data). It is obvious that for obtaining useful results in a reasonable time, the number of processors needs to be much higher.

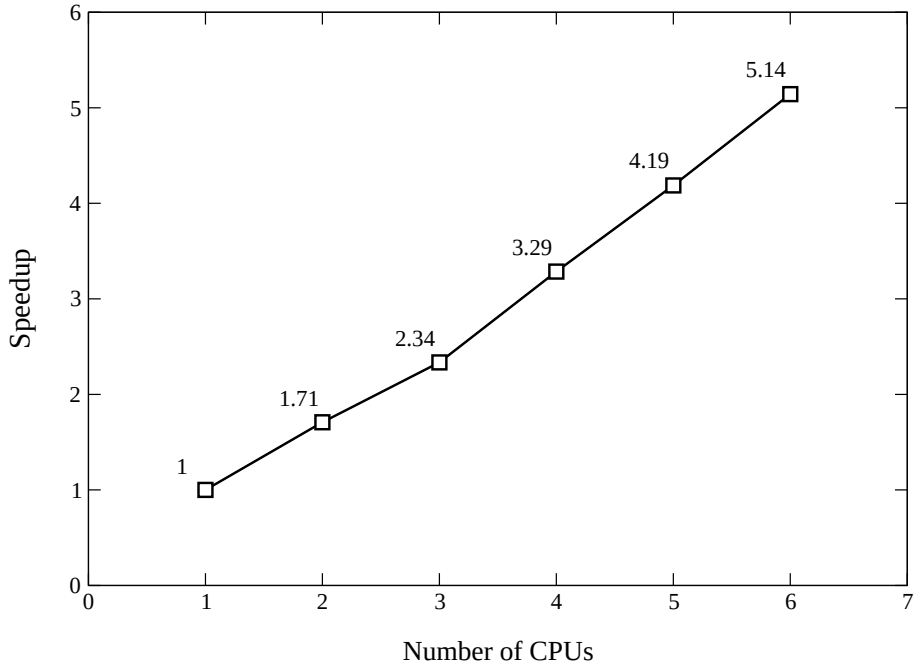


Figure 3: Speedup of the parallel SADE algorithm on a cluster of PCs.

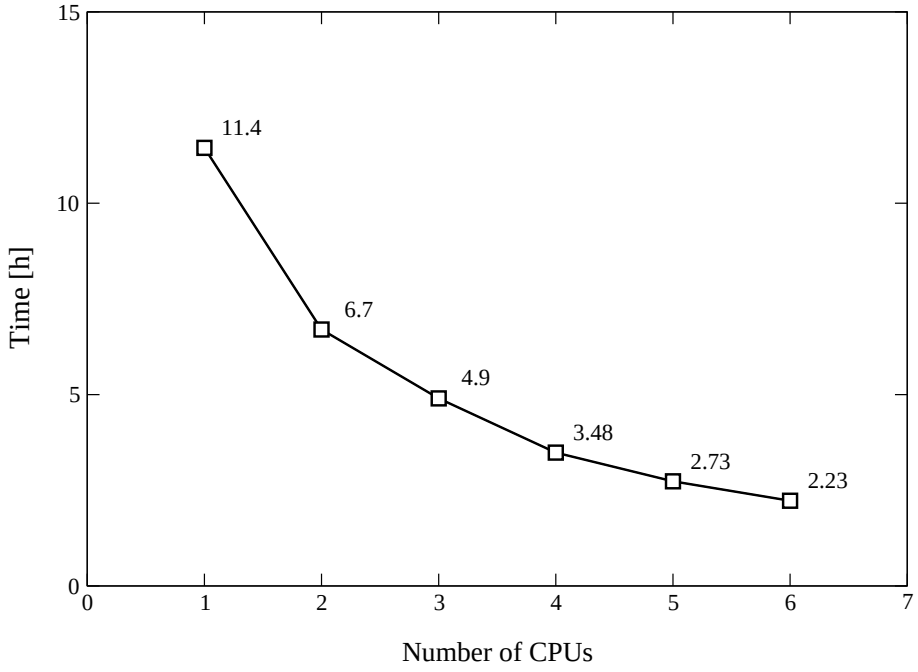


Figure 4: Obtained times during 10 generations of the parallel SADE algorithm.

Processor	No. processors	Memory [MB]
Intel Pentium III 1000 MHz	2	512
Intel Pentium III 1000 MHz	2	2048
Intel Xeon 1700 MHz	2	1024

Table 3: Parameters of used computers.

6 CONCLUSION

In this work, an example of the engineering problem, which is difficult to be solved by traditional procedures, was presented. The application of the stochastic global optimization method SADE brings new possibilities to this area as the optimized function apparently possesses several (at least five) local minima, which introduces considerable obstacles to successful application of gradient-based optimization procedures.

The main outcome of the present study, however, is the conclusion that the determination of the microplane model parameters needs at least two test cases rather than a sole uniaxial compression test. Indeed, this conclusion directly follows from the large scatter of values of Young's modulus E and Poisson's ratio ν among identified local minima (see embedded table in the Figure 2). Since these two parameters are usually the only known values in engineering practice, such a difference is not acceptable and additional data must be supplied to reliably classify individual local minima. Therefore, in this optimization problem, the application of the optimization procedure does produce not only (sub)-optimal values of optimized parameters but also provides deeper insight into the analyzed problem.

The parallel solution then appears to be an appropriate tool how to tackle with enormous computational demand of the microplane material model. Obtained nearly-linear speedup together with possibility to use much more processors promise new interesting results and potential applications of the presented method in the future.

Acknowledgements

The financial support of this work by research project of the Czech Ministry of Education, MSM 210000003, is gratefully acknowledged.

REFERENCES

[1] Z. P. Bažant, F. C. Caner, I. Carol, M. D. Adley, and S. A. Akers, *"Microplane model M4 for concrete. I: Formulation with work-conjugate deviatoric stress"*, Journal of Engineering Mechanics, 126(9):944–953, September 2000.

[2] F. C. Caner and Z. P. Bažant, *"Microplane model M4 for concrete. II: Algorithm and calibration"*, Journal of Engineering Mechanics, 126(9):954–961, September 2000.

[3] E. Cantú-Paz, *"A survey of parallel genetic algorithms (Illi-GAL Report No. 97003)"*, Technical report, Urbana, IL: University of Illinois at Urbana-Champaign, 1997.

[4] E. Cantú-Paz, *"Efficient and Accurate Parallel Genetic Algorithms"*, Kluwer Academic Publishers, 2001.

[5] J. Drchal, A. Kučerová, and J. Němeček, *"Optimizing synaptic weights of neural networks"*, In B.H.V. Topping and Z. Bittnar, editors, Proceedings of the Third International Conference on Engineering Computational Technology, Stirling, United Kingdom, 2002. Civil-Comp Press.

[6] O. Hrstka and A. Kučerová, *"Search for optimization method on multidimensional real domains"*, CTU Report, 4:87–104, 2000.

[7] M. E. Jirásek and Z. P. Bažant, *"Inelastic Analysis of Structures"*, John Wiley & Sons, 2001.

[8] A. Kučerová and O. Hrstka, *"Homepage of SADE"*, `http://klobouk.fsv.cvut.cz/~ondra/sade/sade.htm`.

[9] S.W. Mahfoud and D. E. Goldberg, *"Parallel recombinative simulated annealing - A genetic algorithm"*, Parallel Computing, 21(1):1–28, 1995.

[10] J. Němeček, *"Modelling of compressive softening of concrete"*, CTU Reports, 4(6), 2000. Ph.D. Thesis.

[11] R. Storn, *"Homepage of Differential Evolution"*, `http://www.icsi.berkeley.edu/~storn/code.html`.

[12] R. Storn, *"On the usage of differential evolution for function optimization"*, In NAPHIS 1996, pages 519–523. Berkeley, 1996.