4th International Conference on Inverse Problems in Engineering
Rio de Janeiro, Brazil, 2002

A PARALLEL GLOBAL OPTIMIZATION ALGORITHM FOR SOLVING INVERSE PROBLEMS

Henryk Telega
Institute of Computer Science
Jagiellonian University
Kraków, Poland
telega@ii.uj.edu.pl

ABSTRACT

In this paper two improved versions of *Genetic Clustering* (*GC*) algorithm [1] are described. *GC* is a parallel global optimization algorithm that was designed in order to solve such parameter inverse problems in which an approximation of certain level sets (central parts of basins of attractions of local minimizers) is required. The approximation of these sets can be useful when some additional criteria of optimization are considered after main results of parameter identification are obtained. In spite of some good properties of *GC*, tests have shown that *GC* is not effective for problems with more than 4 dimensions.

Two modifications of *GC* are proposed in order to overcome the dimensionality limitation. In the first modification clusters are remembered as ellipsoids. The second modification is based on the idea of cluster recognition with the use of Kohonen Self Organizing Maps (SOM) neural networks [2].

INTRODUCTION

One of sources of difficulties that are encountered in parameter inverse problems – apart of bad conditioning – is the existence of many solutions. The both mentioned properties make such problems to be ill posed. The paper focuses on parameter inverse problems that are formulated as global optimization tasks. Moreover, the algorithms that are considered are especially suitable for problems, in which an approximation of certain level sets (central parts of basins of attractions of local minimizers) is required. The approximation of these sets can be useful when some additional criteria of optimization are considered after main results of parameter identification are obtained. Such criteria can express in some way for instance the availability and/or the cost of materials. When one knows approximations of central parts of the basins he can give an approximate answer to the question: how much one can change the value of a parameter with "not too high" change of the objective.

A hybrid genetic parallel algorithm called *GC* (*Genetic Clustering*) was proposed in [1] in order to solve such problems.

The *GC* strategy is inspired with clustering methods in global optimization [3], [4], [5] and genetic algorithms [6]. *GC* finds local minimizers and also gives additional information – central parts of basins of attraction of local minimizers can be approximated.

PARALLEL GENETIC CLUSTERING (*GC*)

The aim of original version of *GC* algorithm is to find all local minima that have adequately large basins of attraction with a sufficiently large objective variability. The algorithm also gives rough information about the basins. The basins (or their central parts) are approximated by sets of hypercubes – these sets will be called clusters. Additionally, the algorithm deals with large differences between values of local minima and also with large plateaus.

An outline of a version of the *GC* algorithm is recalled below. Some asymptotic properties of the algorithm have been derived from the Markov theory of the Simple Genetic Algorithm [7]. The stop criterion has been justified in [8].

The genetic approach to clustering consists in implementing Simple Genetic Algorithm (SGA) in the global phase [3]. The idea of sampling by running SGA follows from the observation that genetic algorithms transform measures in a regular way so they deliver information about sets rather than about isolated points. In its nature

Genetic Algorithm constitutes a dynamical system that transforms measures. This fact allows us to expect good properties of genetic clustering, because density of measure contains information that is useful in clusters recognition.

GC consists in four operations that are executed consecutively: genetic sampling, subclusters' recognition, subclusters' aggregation and fitness modification. These operations are performed in a loop until the global stop criterion is satisfied.

The outline of the *GC* is as follows:

```
1. Divide the feasible set $D$ into $p$ subdomains
   (each subdomain is divided into small
   hypercubes that constitute the grid).
2. Set all subdomains to be "active".
3. REPEAT
Parallel in "active" subdomains:
    3.1 Generate initial population from uniform
        distribution.
    3.2 Evaluate fitness function $f$ outside
        recognized clusters.
    3.3 Modify fitness function ($f \leftarrow MAX$ on
        clusters).
    3.4 Steps of simple genetic algorithm (SGA)
        - evaluate new generations until the
        complex stop criterion is satisfied:
        a) subclusters can be recognized, or
        b) GA recognizes plateau outside
           known clusters (then the subdomain
           is set to be "passive").
        Subclusters are parts of clusters that can
        be recognized after point 3.4.
    3.5 Subclusters recognition. (output: new
        information about clusters and new
        "passive" subdomains). The seed of a
        subcluster is this cell (hypercube) of the
        grid that contains "the best" individual.
        All neighbor cells that contain more
        individuals than a certain threshold are
        added to the subcluster. One local
        method is started in each subcluster.
    3.6 Join "proper" subclusters into clusters.
UNTIL all subdomains are "passive" OR
satisfactory set of clusters is found.
```

The Simple Genetic Algorith (*SGA*) was chosen in the genetic phase of *GC*, because it allowed us to obtain some theoretical results concerning stop criterion and asymptotic behavior of *GC*.

The fitness modification results in repelling individuals from clusters (subclusters) that are already known. A single basin of attraction can be recognized in one or several steps of the loop. The domain *D* is divided into hypercubes of a volume θ. After the *SGA* is stopped, new subclusters can be detected by the analysis of density of individuals in the hypercubes. The hypercube that contains the best individual is selected as the seed of a new subcluster. Neighbor hypercubes, with the density of individuals $\rho > \rho_t$ (ρ_t is an arbitrary constant), are attached to the cluster. A rough local optimization method is started in each new subcluster and the result of this optimization is retained. If the local method that starts from a new subcluster ends in the already recognized cluster, then the subcluster is attached to the cluster.

The stop criterion distinguishes two basic kinds of *SGA* behavior. The first one is that *SGA* "finds" clusters after few generations. The second one is that *SGA* converges to the uniform distribution of individuals. This corresponds to the recognition of a plateau (or areas where fitness has small variability) outside of the already known clusters. Other cases are treated as the situation when *SGA* does not fit to the particular problem and a refinement of *SGA* parameters is suggested.

The stopping strategy is as follows:

Check stagnation of a sequence of some estimator of probabilistic distribution density. If this criterion is satisfied, then check if an arbitrary number of hypercubes has the density of individuals below an arbitrary threshold ρ_t. If so, then begin clustering procedure, otherwise, check if individuals are uniformly distributed in *D*.

In original version of *GC* each subdomain is divided into hypercubes that constitute a static grid. Each cell of the grid that belongs to some cluster "remembers" the number of the cluster.

PROPERTIES OF *GC*

Each population with a finite number of binary coded individuals can be identified with a vector which *i*-th coordinate stands for the occurrence frequency of the *i*-th individual in the population. Lets by *r* denote the length of an individual. The frequency vector belongs to the unit simplex Λ in $\Re^{r-1}$. All possible populations of the size n correspond to the finite subset S_n in Λ [6].

The finite population SGA constitutes a stationary Markov chain with states from S_n. By non-zero mutation it is ergodic, and there exists a weak limit

$$\pi_n^k \underset{k\to\infty}{\overset{w}{\to}} \pi_n \tag{1}$$

of probability distributions π_n^k on S_n in k-th generation [6].

In the case of an infinite population $n=\infty$ *SGA* is a deterministic dynamic system with states in Λ governed by the genetic search operator $\Gamma:\Lambda\to\Lambda$. The sequence of the limit probability distributions π_n has a weak limit distribution π^* when the size of population goes to infinity $n\to\infty$. Moreover if Γ is focused, and K is its set of fixed points then $\pi^*(K)=1$ [6].

Let F_ε the ε-envelope of the set K

$$F_\varepsilon = \{x\in\Lambda; \quad \exists y\in K; \quad d(x,y)<\varepsilon\} \tag{2}$$

where d stands for the distance function in $\Re^{r-1}$.

Lemma [8]*:* $\forall\varepsilon>0\ \forall\varsigma>0\ \exists N>0$ such that $\forall n>N\ \exists G(n)$ and $\forall k>G(n)\ \pi_n^k(F_\varepsilon)>1-\varsigma$.
It means, that if the population is sufficiently large and a sufficiently large number of generations were performed, then the population is arbitrary close to the fixed one with the arbitrary large probability.

It is assumed here that SGA parameters are so chosen that Γ is focused. In particular small but non-zero mutation is assumed. The desired form of the fixed points set is the finite collection of isolated points in Λ. Moreover each local minimizer of the fitness function is represented in K. It corresponds to the population highly concentrated on its neighborhood.
Conjecture: only minimizers that have sufficiently large attractors (larger than the cell size) with the sufficiently high fitness variation can be found.

Algorithm detects situation in which the population is sufficiently concentrated in attractors so that the density cluster recognition is possible. The state in which arbitrary rate of grid cells contain the assumed number (less than the average) of individuals can be handled as the local stop criterion. By Lemma the above situation is asymptotically highly probable if there exists at least one attractor out of the cluster union.

The chart of modified fitness function becomes sufficiently flat at the end of computations. It corresponds to the unique fixed point of Γ at the center of Λ.

If the sufficiently large population that starts from the center of Λ (uniform distribution of individuals) does not leave its neighborhood sufficiently long this implies that the center of Λ is the fixed point of Γ (with the arbitrary large probability). It corresponds to the situation, that the probability of finding new local minimizers is arbitrary small.

One can say, that there is an analogy between the way in which mutation and crossover rates in *SGA* imply *GC* algorithm and the way in which the reduction phase implies *DC* and *SL* clustering algorithms described in [4], [5]. Both factors cause that some minima can be undetected. However, unlike the *DC* and *SL* with the reduction phase, the GA constitutes a filter that eliminates local minima with small fitness variability and shallow basins of attraction. GA strategy is also less sensitive on fitness values in local minimizers. Such filtering property can be useful in some cases. Another interesting feature of *GC* is such that it should be especially convenient for functions with large areas of small variability (areas "similar to" plateaus) which can be difficult for other methods.

Genetic clustering offers some interesting properties like:

- Approximation of basins for all "remarkable" local minimizers,
- Filtering local minima with sufficiently small and "shallow" attractors,
- Good time complexity in cases of objectives with large plateaus.
- The global stop criterion of this strategy can be mathematically justified; this is rarely met in the case of strategies based on evolutionary algorithms.

Tests described in [1], [8] have shown that *GC* can be effective in solving some inverse problems including for instance the problem of optimal pretraction design of a network structure made of elastic unconnected fibers fastened at their ends to a square rigid frame.

However, *GC* is not effective for problems with more than 4 dimensions. This follows from the representation of clusters – they are remembered as unions of small hypercubes that constitute a regular mesh in the domain of searches.

In order to overcome the above limitation three modifications to the *GC* algorithm are proposed. They will be described in following sections.

CLUSTERS REPRESENTATION WITH THE USE OF ELLIPSOIDS

One of methods that can be proposed to overcome the problem with high dimensionality is to represent clusters by ellipsoids. The similar approach to clusters in global optimization is known in so called *Density Clustering* (*DC*) rule described in [3]. Some good properties of this version of *DC* are proven in [5]. The version of *DC* proposed by Rinnooy Kan and Timmer assumes that the reduction phase is applied, that means the initial sample is drawn from the uniform distribution over *D* and all sample points where the value of the objective function is below certain threshold are rejected [5]. A key assumption is that the objective function is well approximated by a quadratic function in neighborhoods of local minimizers. This implies that level sets (and clusters) are approximated by ellipsoids. Clusters are recognized iteratively in the following way: the seed point $\bar{x}$ of a cluster is the result of local optimization started from the unclustered best point of the reduced sample (the unclustered point with the smallest value of the objective function). Lets by T_0 denote the set { $\bar{x}$ } with the seed of the cluster. In consecutive steps next points of the sample are joined to the cluster. These points belong to subsets T_i of *D*, *i=1,2,...*, $T_i \subset T_{i+1}, i = 1,2,...$. These subsets correspond to certain level sets. When $f \in C^2$ we can approximate level sets by

$$T_i = \left\{ x \in D | (x-\bar{x})^T H(x_s)(x-\bar{x}) \le r_i^2 \right\},$$

where *H* denotes hessian.

All points that are within T_i which is described by a critical distance $r_i(\bar{x})$ of the seed are joined to the cluster. The distance $d(x_1, x_2)$ is defined as follows: for points x_1, x_2 from a neighborhood of $\bar{x}$

$$d(x_1, x_2) = \left[(x_1 - x_2)^T H(\bar{x})(x_1 - x_2) \right]^{\frac{1}{2}},$$

(an approximation of hessian can be obtained as a byproduct of quasi-Newton local methods). The parameter $r_i(\bar{x})$ is increased stepwisely (with increasing *i*) until there is no unclustered point from the reduced sample within $r_i(\bar{x})$. Rinnooy Kan and Timmer gave the formula for the critical distance:

$$r_i(\bar{x}) = \pi^{-\frac{1}{2}} \left(i\Gamma\left(1+\tfrac{d}{2}\right) \det(H(\bar{x}))^{\frac{1}{2}} m(D) \frac{\sigma \log kN}{kN} \right)^{\frac{1}{d}}, \qquad (3)$$

where Γ is the Gamma function, *m* denotes the Lebesgue measure, *N* is the sample size and σ is a constant. The whole process of sampling, reduction and clusters recognition is repeated (*k* denotes the number of the iteration) until a global stop criterion is satisfied. The formula (3) assures that the probability of erroneous termination of the cluster recognition procedure (the procedure is terminated too early, see [5] for details) in step *i* decreases polynomially fast with increasing *k*.

This version of *DC* has also other advantages:

- It has the property of asymptotic correctness in the sense that it finds global minimum with the probability 1 as *k* increases to infinity.
- It is possible (and relatively easy) to apply bayesian stopping rules [3], [5].

The main drawbacks are obvious:

- The success of the method depends on how well the assumed approximation is.
- In fact each recognized cluster can contain more than one minimizer.

Applying similar approach to *GC* can diminish disadvantages that are caused by high dimensionality. Clusters are parametrized by the central point and radiuses. Each point generated by *SGA* can be classified as belonging to some cluster or not, so the idea of fitness modification can be almost unchanged.

The Bayesian stopping rules derived for *DC* cannot be applied directly to *GC*, because these rules assume uniform distribution of sample points. Also such good properties of *DC* as mentioned above cannot be directly attributed to *GC*. Analogous estimations for *GC* are still open problems, because it is difficult to predict and calculate the exact distribution of points after some genetic epochs.

Introducing such cluster representation to *GC* causes also that stopping strategy from *GC* should be modified. Under the assumption that clusters cannot intersect, the criterion "all subdomains are passive" in real cases should be removed.

The critical distance $r_i(\overline{x})$ has not the same meaning as in *DC*, but it can be probed as if the concentration of points would be caused by the reduction phase with sample points distributed uniformly.

CLUSTERS RECOGNITION AND REPRESENTATION WITH THE USE OF NEURAL NETWORKS

Clustering methods are known also as methods that help in constructing categories or taxonomies. Special kind of neural networks – so called self-organizing maps *SOM* ([2] and bibliography cited there, [9]) were proposed as tools for exploratory data analysis, in particular for visualization of high dimensional data items.

We propose to join *GC* with the mechanism analogous to *SOM* in order to recognize and remember clusters. Additionally the method can visualize clusters in some way.

SOM is a special kind of neural network with competitive learning. Competitive learning is an adaptive process in which the neurons gradually became sensitive to different input categories [2], for instance clusters of points. After the process of learning is finished, neurons become specialized – they "represent" different categories (clusters). The mechanism which allows neurons to specialize bases on a competition among them. After an input data x arrives, this neuron wins which better "represents" the data. Moreover neurons can "learn data" (it will be described below).

In *SOM* neurons are located on a discrete lattice that constitute the "self-organizing map". During the learning process the winning neuron and its neighbors on the lattice are allowed to learn.

The input data is represented in neurons by a vector w_i (reference vector), whose components correspond to synaptic weights. Neurons can be indexed with k. The winner neuron is determined from the formula:

$$k = k(x) = \arg\min_i \left\{ \|x - w_i\|^2 \right\} \tag{4}$$

That means the winner is this neuron, whose reference vector is closest to the input data x. This neuron and its neighbors modify their reference vectors according do the formula (5).

$$w_i(t+1) = w_i(t) + h_{ki}(t)[x(t) - w_i(t)] \tag{5}$$

Neighbors are determined by so called neighborhood kernel function h_{ki}.

In the simplest case the neighborhood function can be defined as follows:

$$h_{ij} = \begin{cases} 1 & \|r_i - r_j\| \le \lambda \\ 0 & \|r_i - r_j\| > \lambda \end{cases} \tag{6}$$

or

$$h_{ij} = \exp\left(-\frac{\|r_i - r_j\|^2}{2\lambda^2} \right) \tag{7}$$

where r_i and r_j are vectors that represent location of neurons in the lattice, t denotes time and λ is a constant.

In general the neighborhood function can also be variable in time – "wide" at the beginning of the learning process and decreasing slowly during learning.

In such approach clusters can be represented as reference vectors of neurons. The number of clusters does not need to be estimated in advance, the maximum number is equal to number of neurons. Learned neurons can categorize any further sample points to clusters.

Additionally a method of visualization of clusters was proposed by Kohonen [9]. The distances between the reference vectors of neighbor neurons can visualize the clusters structure on a two dimensional map. Details can be found in [2].

TESTS

The modified version of the *GC* in which clusters were represented as ellipsoids was tested on the same problems as the original version of *GC* (see [1], [8]).

The representative results of these tests will be presented for the 8-dimensional global optimization problem given by formula (9). Also the 16-dimensional version was tested. The original version of *GC* was tested on the 2-dimensional case of this problem (see [1]). A parameter inverse problem with a similar cost function and analogous complexity was presented in [1]. It was optimal pretraction design of a network structure made of elastic unconnected fibers fastened at their ends to a square rigid frame.

The linearized and homogenized governing equation (see [1]):

$$\begin{cases} -\operatorname{div}(\sigma\, Dw) = q & \text{in } \Omega, \\ w = 0 & \text{on } \partial\,\Omega \end{cases} \tag{8}$$

delivers the relationship between the compliance $w(x)$, pretraction tensor $\sigma(x) = diag(\sigma_1(x_2), \sigma_2(x_1))$, and the transversal loading density $q(x)$ on the frame area Ω.

Given q try to find $\sigma^* \in (L^\infty(\Omega))^2$ such that $\mathrm{F}(\sigma^*) \le \mathrm{F}(\sigma)$ for all $\sigma \in (L^\infty(\Omega))^2$, and $w_{\sigma^*} \in H_0^1(\Omega)$ satisfies the state equation (8).

The cost functional $\mathrm{F}(\sigma) = \mathrm{E}(\sigma) + \mathrm{P}(\sigma)$, where $\mathrm{E}(\sigma) = \int_\Omega \sigma |Dw_\sigma|^2 dx$ is the stored energy of the network, and $\mathrm{P}(\sigma)$ denotes a penalty that forces pretractions suitable for an available assortment of fibers. $\mathrm{P}(\sigma)$ is a multimodal, nonnegative function which reaches zero for many admissible pretractions.

Moreover, the following constraints are defined: $0 < \lambda \le \sigma_1, \sigma_2 \le \Lambda$ in Ω, $\int_\Omega (\sigma_1 + \sigma_2) dx = S$ with $S \in [2\lambda, 2\Lambda]$.

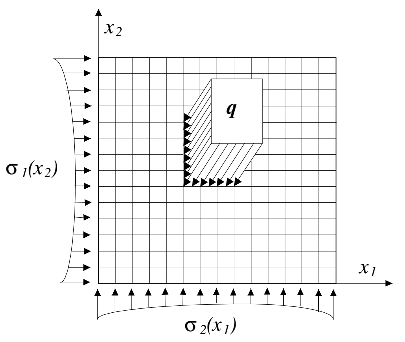

Figure 1. Schema of the fiber construction

The case of a balanced loading $\int_\Omega q dx = 0$ was considered. It is proved that under above assumptions there is more than one minimizer σ^*.

The adequate test global optimization problem can be as follows (see [1]):

$$f(x) = \sum_{i=1}^{8} 0.01 \sin(0.05 x_i) + \\ \text{-c*} \begin{pmatrix} x_j > 60\ AND \\ x_j < 70\ j = 1,\ldots 8 \end{pmatrix} * \sum_{i=1}^{8} x_i^2 + \\ \text{-c*} \begin{pmatrix} x_{2j+1} > -50\ AND \\ x_{2j+1} < -20\ AND\ j = 0,1,2,3 \\ x_{2j+2} > -70\ AND \\ x_{2j+2} < -40 \end{pmatrix} * \\ * \sum_{i=1}^{8} x_i^2 \tag{9}$$

where $f : R^8 \to R$, x_i, i=1,…8 stand for the components of x. The formula $(x_j > a\ AND\ x_j < b\ , j = \ldots)$ stands for one, if the condition in brackets is true for all given j, otherwise it stands for zero.

The constant c for 8-dimensional problem was equal to 0.00024. The domain of searches was a hypercube given by

$$-100 < x_i < 100\ , i=1,\ldots,8 \tag{10}$$

The function f has two distinct "deep" local minima and many "shallow" local minima. One of the deep minima is the global minimum. Two-dimensional version of f is presented on Figure 1.

It is assumed that clusters are well approximated by spheres. For the 8-dimensional case the population of SGA was 500. Each time twenty generations were processed before clustering was applied. Only two distinct "deep" minima were discovered and two clusters were located.

Table 1. Results for 8-dim. problem

	minimum 1 *f(x)*=5.11	**minimum 2 *f(x)*=2.61**
component of *x*	**value**	**value**
1	-50,0	70,0
2	-70,0	70,0
3	-50,0	70,0
4	-69,7	69,9
5	-49,9	69,7
6	-66,5	69,8
7	-49,9	69,5
8	-66,6	69,5

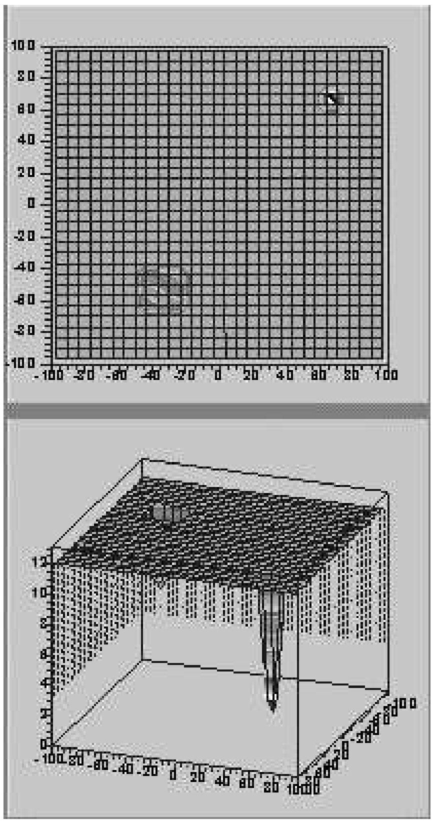

Figure 1. Two dimensional version of the 8-dimensional test function.

The scalar version of the algorithm was applied (without the division of the domain).

The results are presented in Table 1. The radius of the cluster 1 (minimum 1) was 32.2, the radius for the cluster 2 (minimum 2) was 8.37. The number of function calls in local searches (MIGRAD method from the CERN ROOT-Minuit package) was 1752 (968 for cluster 1 and 784 for cluster 2).

These results are much better than for original version of *GC* in which a great number of cells must be considered (this number depends of the chosen resolution).

However, clusters are not recognized so precisely as in *GC*. They do not contain whole basins of attraction of local minima. Moreover, they also include parts of the domain that should not be included. A modification is required when one would like to use points from clusters. For each such point the value of the objective function should be calculated (in tests the maximum value of the objective function was also remembered for each cluster).

When one wants to recognize clusters more precisely he or she can apply the original version of *GC* with the smaller domain that includes recognized cluster/clusters.

Tests showed, that the proposed algorithm maintains the filter property.

The maximum 16-dimensional case was tested successfully, however this is not the largest possible dimension. It seams, that problems can occur with the local strategies when the dimension is bigger than several tens.

The version with neural network representation of clusters is being tested now.

First tests made for two-dimensional case of the presented problem have shown that this method of clustering data in *GC* is costly in time and its superiority could be seen for problems with not too low dimensionality. One of ways to accelerate computations is to parallelize algorithm that simulates neural networks, another is to construct a hardware neural network.

ACKNOWLEDGMENTS

Research supported in part by the State Committee for Scientific Research of the Republic of Poland (KBN) under Grant No. 7 T07A 047 18.

REFERENCES

1. H. Telega, *Adaptive Parallel Genetic Clustering in Parameter Inverse Problems*, Inverse Problems in Engineering Mechanics III, M. Tanaka, G. S. Dulikravich Eds., Elsevier 2002, p. 315-325.
2. S. Kaski, *Data Exploration Using Self-Organizing Maps*, Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82, 1997.
3. R. Horst, Pardalos M. Panos, *Handbook of Global Optimization*, Kluwer Ac. Publisher 1995
4. A.H.G. Rinnooy Kan, G. T. Timmer *Stochastic global optimization methods. Part I: Clustering methods,* Mathematical Programming **39**, 1987, p. 27-56.
5. A. H. G. Rinnoy Kan, G. T. Timmer, *Stochastic global optimization. Part 2: Multi level methods*. Mathematical Programming **39**, 1987, p. 57-78.
6. M. D. Vose, The Simple Genetic Algorithm - Foundation and Theory. The MIT Press, 1999.
7. Cabib E., Schaefer R., Telega H. [1998] A parallel genetic clustering for inverse problems.

Lecture Notes in Computer Science **1541**, Springer 1998, p. 551-556.

8. H. Telega, PhD Thesis, AGH Krakow, Poland (in polish), 1999.

9. T. Kohonen, *Self Organizing Maps*, Springer, Berlin, 1995.