JOURNAL OF THEORETICAL
AND APPLIED MECHANICS
42, 3, pp. 519-538, Warsaw 2004

HIERARCHICAL GENETIC COMPUTATION IN OPTIMAL DESIGN

Joanna Kołodziej

Institute of Mathematics, University of Bielsko-Biała
e-mail: jkolodziej@ath.bielsko.pl

Robert Schaefer
Anna Paszyńska

Institute of Computer Science, Jagiellonian University
e-mail: schaefer@ii.uj.edu.pl; paszynska@ii.uj.edu.pl

The paper presents two examples of genetic hierarchic global optimization methods. They are two different goals of introducing hierarchy into the computational model: to perform the multi-scale search with the adapted accuracy and to better express the structure geometry in the optimal shape design. Results of the formal analysis and simple computational examples are also attached.

Key words: hierarchical genetic optimization, graph encoding, shape optimization

1. Introduction

1.1. Hierarchy in biology and mechanical engineering

We may distinct two goals in hierarchy creation or identification in optimal design problems coming from mechanical engineering:

- To select the proper scale or the accuracy of analysis. A whole, large civil structure may be analyzed as a single unit if the wind resistance is analyzed. The plate being one of its part may be handled as a distinct structure in the smaller scale and may be analyzed as a complicated, periodic construction if its internal organization should be improved. The opposite passage may be observed if we analyze a composite material in the scale of its one characteristic cell (unit) and then go to the macro-scale analysis by homogenizing the parameters of components.

- To express the internal structure or the method of manufacturing. The feature-oriented representation of the geometry of mechanical parts allows one to represent the geometry as the log of the manufacturing process. The root of this hierarchy may be interpreted as a crude piece of material and the low level objects as effects of the particular, dependent tooling actions.

We may also underline some hierarchical dependencies within biological systems.

- Hierarchy among body parts of individuals caused by the cellular structure, tissues and organs follows the hierarchy of the individual body structure.
- Hierarchy of individuals imposed by the degree of complexity of species. Usually simple, low complex species (microbes, plankton, insects,...) explore first new environment regions finding suitable niches. If the environment is partially recognized, then high complexity species can obtain the maximum fitness in the selected parts of niches.

Hierarchy identified in mechanical systems encourage us to emply hierarchic computational methods. We may distinct a group of genetic algorithms that follows hierarchy of a computational problem by utilizing the proper biological metaphor.

1.2. Short taxonomy of genetic hierarchical methods

The first group of algorithms introduces the hierarchy of accuracy in the global genetic searches performed in continuous domains. Delta coding (see Whitley *et al.*, 1991) introduces hierarchy of binary codes that allow more intensive search around the best fitted individual encountered previously. The genotype shortening is performed by keeping the prefixes at the value characteristic for the best individual. This strategy is also extended to the multi-population one based on the island scheme (see Mathias, 1991). The hierarchy of searches is utilized in the Dynamic Parameter Encoding DPE (see Whitley *et al.*, 1991). The admissible domain is spread into the nested system of subdomains. If the single-population binary coded genetic algorithm is dense in one ore more subdomains, the search will be restricted to these parts and "zoomed" by recoding all genotypes to the new smaller domain. Another idea is represented in the Optimal Hierarchic Algorithm OHA described by Cant-Paz (see Bickel and Bickel, 1987). A two-level hierarchy is introduced. The island structure of many populations makes a higher level. The fine-grained parallel implementation of each island population combined with the master-slave or cellular parallel mechanisms is used at the lower level of hierarchy. The most comprehensive strategy that creates a hierarchy of parallel dependent robust searches with the different resolution HGS will be described in Section 2.

The second group of genetic algorithms follows the internal structure of the mechanical part during optimization. Hierarchical structures can be represented as a hierarchical graph being a genotype in graph-oriented genetic algorithms (see Nikodem, 2000; Wierzba *et al.*, 2003). Graph-based genetic operators are usually more complex than binary ones but the benefits coming from the possibility of coding relationships between the components of an artifact and ability to introduce structural alternations compensate it. Other solution is a modified hierarchical chromosome with the corresponding crossover and mutation operators (see Paszyńska, 2002). This representation is easier to implement than the graph-based one, but no direct information of relations between primitives of the artifact is available. The advantage of the hierarchical chromosome based method is a possibility to start an evolutionary process with a random population. In the graph-based one it is an ability to produce highly fit individuals faster, but the user-provided initial population is required. An example of such an approach is contained in Section 3.

Hierarchical structures are often a representation of individuals in many other evolutionary algorithms. Already Wilson wrote in the year 1987 (see Wilson, 1987) about the meaning of hierarchy in the representation of tasks and subtasks. In the same year, Bickel A.S. and Bickel R.W. (see Bickel and Bickel, 1987) used trees in their system. Also the genetic programming is often based on trees (see Koza, 1992; and Langdon and Poli, 2002). In spite of numerous applications, there is still little known about the mathematical model and theoretical results of evolutionary algorithms based on hierarchical structures.

2. Hierarchical Genetic Strategy (HGS) for the continuous parameter optimization

2.1. The idea of HGS search

The Hierarchical Genetic Strategy (HGS), introduced by Schaefer *et al.* (2000), is a kind of multideme, parallel genetic algorithm, which is a very effective tool in solving ill-posed problems of continuous global optimization with multimodal and weakly convex objective functions. High efficiency of the strategy comes from the concurrent search in the optimization landscape by many small populations.

The main idea of the Hierarchical Genetic Strategy (HGS) is running in a parallel set of dependent evolutionary processes. The dependency relation among processes has a tree structure with the restricted number of levels. The processes of lower order (close to the root of the structure) represent chaotic search with a low accuracy. They detect promising regions in the optimization landscape, in which more accurate processes of higher orders are activated. Every process creates a branch of the tree and can be defined as a sequence of evolving populations. Populations evolving in different processes can contain individuals which represent the solution (the phenotype) with different precision. This precision can be achieved by binary genotypes of different lengths or by different phenotype scaling.

The strategy starts with the process of the lowest order called the root. After a fixed number of evolution epochs, the best adapted individual is selected. We call this procedure the *metaepoch* of the fixed period. After every metaepoch, a new process of a higher order can be activated. This procedure is called the *sprouting operation.* It is performed conditionally according to the outcome of the *branch comparison operation*, and can be generalized to some extent to HGS branches of higher orders. The definitions of those operations depend strongly on the implementation of the HGS. In the binary implementation of the HGS (see Schaefer and Kołodziej, 2003) the Simple Genetic Algorithm (SGA) was applied as *the law of evolution* in every process. The replacement of the SGA engine by a mechanism based on real-number encoding, normal mutation and the simple arithmetic crossover increases the efficiency of the HGS (see Wierzba *et al.*, 2003).

2.2. Binary implementation of the HGS

Let us define the basic object and operations in the binary implementation of HGS.

Let us denote by Ω_s a set of all possible binary coded genotypes of the fixed length $s \in \mathbb{N}$. A collection of n elements of Ω_s is called *a population of the size* n and can be represented by its *frequency vector*

$$\boldsymbol{p} = [p_0, ..., p_{r-1}]^\top \qquad p_j \geqslant 0 \qquad \sum_j p_j = 1 \tag{2.1}$$

where p_j is the proportion of the element j in the population. The coordinates of this vector are identical with baricentric coordinates of some point in the standard unit $(r-1)$-dimensional $(r = 2^s)$ simplex Λ. We denote by X^r_n a set of all possible populations of the size n.

To start the strategy we have to fix the following parameters

- $1 \leqslant s_1 < s_2 < ... < s_m < \infty$ – lengths of binary coded strings ($\Omega_{s_1}, \Omega_{s_2}, ..., \Omega_{s_m}$ are the genetic spaces associated with those strings)
- $1 \leqslant n_1 \leqslant n_2 \leqslant ... \leqslant n_m$ – sizes of populations which are multisubsets of $\Omega_{s_1}, \Omega_{s_2}, ..., \Omega_{s_m}$, respectively.

We can say that a given process has an order j, $(i \in \mathbb{N})$, if the individuals from the evolving populations have genotypes of the length s_j. The unique branch of the lowest degree 1 is called *the root*. The populations evolving in this branch contain individuals with genotypes of the shortest length.

The *nested coding* is a method of coding the designed for the HGS with SGA genetic engine (see Schaefer and Kołodziej, 2003). This method was first defined for $D \subset \mathbb{R}$ and then generalized to $D \subset \mathbb{R}^N$, $N > 1$, $\mu(D) > 0$, where μ stands for the Lebesgue measure over $\mathbb{R}^N$. For $D \subset \mathbb{R}$ we have to define recursively the sequence of meshes $D_{r_1}, ..., D_{r_m}$ and the sequence of strictly increasing coding mappings $code_j$: $D_{r_j} \ni x_{(\omega)} \to \omega \in \Omega_{s_j}$, $j = 1, ..., m$. We start from the densest mesh $D_{r_m} \subset \mathbb{R}$ and the coding function $code_m$: $D_{r_m} \to \Omega_{s_m}$. For every $1 \leqslant j \leqslant m-1$ we define the *selecting function* ϕ_j: $\Omega_{s_j} \to \Omega_{(s_{j+1}-s_j)}$ which is fundamental for the construction of the sets D_{r_j}, $j = 1, ..., m-1$. We put

$$D_{r_j} = \{x_{(\omega,\phi_j(\omega))},\ \omega \in \Omega_{s_j}\}$$

where $x_{(\omega,\xi)} = code^{-1}_{j+1}(\omega,\xi)$, $1 \leqslant j \leqslant m-1$.

For $D \subset \mathbb{R}^N$, $N > 1$ and for $\overline{s}_j \in \mathbb{N}$, $N\overline{s}_j = s_j$ we define the sets $D^1_{r_m}, ..., D^N_{r_m} \subset \mathbb{R}$ such that the Cartesian product $D^1_{r_m} \times \ldots \times D^N_{r_m}$ is included in D. We also define the strictly increasing mappings $code^i_m$: $D^i_{r_m} \to \Omega_{\overline{s}_m}$ for $i = 1, ..., N$. For each $i \in \{1, ..., N\}$ we define recursively a sequence of sets $D^i_{r_j}$, a sequence of mappings $code^i_j$: $D^i_{r_j} \to \Omega_{\overline{s}_j}$, $j = 1, ..., m-1$ and the selecting function ϕ_j: $\Omega_{\overline{s}_j} \to \Omega_{\overline{s}_{j+1}-\overline{s}_j}$ for $1 \leqslant j \leqslant m-1$ such that:

(1) $D^i_{r_j} = \{x_{(\overline{\omega},\phi_j(\overline{\omega}))}, \overline{\omega} \in \Omega_{\overline{s}_j}\}$, where $x_{(\overline{\omega},\overline{\xi})} = (code^i_{j+1})^{-1}(\overline{\omega},\overline{\xi})$, $1 \leqslant j \leqslant m-1$

(2) $code^i_j$: $D^i_{r_j} \ni x_{(\overline{\omega},\phi_j(\overline{\omega}))} \to \overline{\omega} \in \Omega_{\overline{s}_j}$, $1 \leqslant j \leqslant m-1$.

We put:

$$\begin{aligned} D_{r_j} &= D^1_{r_j} \times \ldots \times D^N_{r_j} && j = 1, ..., m \\ \Omega_{s_j} &= (\Omega_{\overline{s}_j})^N && j = 1, ..., m \\ code_j &= (code^1_j, ..., code^N_j) && j = 1, ..., m \end{aligned}$$

Note that $D_{r_1} \subset D_{r_2} \subset ... \subset D_{r_{m-1}} \subset D_{r_m}$. Figure 1 shows the sets D_{r_1}, D_{r_2} and $D_{r_3} \subset \mathbb{R}$ in the case of $s_1 = 2$, $s_2 = 3$, $s_3 = 5$, $\phi_1(00) = \phi_1(01) = 1$, $\phi_1(10) = \phi_1(11) = 0$, $\phi_2 \equiv 01$.

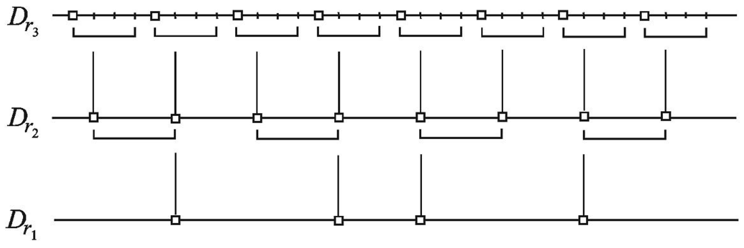

Fig. 1. Nested meshes for the Hierarchical Genetic Strategy in the case $s_1 = 2$, $s_2 = 3$, $s_3 = 5$

The fitness function in the HGS should be defined separately for each branch of degree $j \in \{1, ..., m\}$

$$f_j = \widetilde{\Phi} \circ code_j^{-1} : \quad \Omega_{s_j} \to \mathbb{R}_+ \tag{2.2}$$

One of the basic procedures in the strategy is the *metaepoch*. Formally, we can define the k-periodic metaepoch M_k, $k \in \mathbb{N}$ as a discrete evolution process which starts from the given population and terminates after at most k generations by selection of the best adapted individual. The outcome of the metaepoch starts from the population p^r in the r-th generation. It may be denoted by $M_k(p^r) = (p^{r+l}, \widehat{\omega}, stop)$, $l \leqslant k$, where p^{r+l} denotes the frequency vector of the resulting population, $\widehat{\omega}$ – the best adapted individual in the metaepoch and *stop* – the branch stop criterion flag. The *branch stop criterion* detects the lack of progression in the evolution process. It is usually heuristic (e.g. detects a small increment of the average fitness).

By application of the *sprouting operator* the structure of the HGS can be extended by creating new branches from the given one. To define the sprouting operator, we have to introduce an operator $A_{s'}$ which 'cuts off' an s'-length prefix from the given binary string of the length s'', $s'' > s'$.

Definition 2.1

- *If $D \subset \mathbb{R}$, then for a binary string ω_2 of the length $|\omega_2| = s''$ we have $A_{s'}(\omega_2) = \omega_1$ where $|\omega_1| = s'$; $s'', s' \in \mathbb{N}$, $s'' > s'$. It means that ω_1 constitutes the s'-length prefix of the string ω_2.*
- *If $D \subset \mathbb{R}^N$, $N > 1$, $\mu(D) > 0$, then for $\overline{s}', \overline{s}'' \in \mathbb{N}$ such that $N\overline{s}' = s'$, $N\overline{s}'' = s''$ and for $\omega_j = (\omega_j^1, ..., \omega_j^N)$; $j = 1, 2$; $|\omega_1^i| = \overline{s}'$;*

$|\omega_2^i| = \overline{s}''$; $i = 1, ..., N$ then $A_{s'}$ is defined as the catenation of operators $A^i_{\overline{s}'}$, $i = 1, ..., N$

$$A_{s'} = (A^1_{\overline{s}'}, \dots, A^N_{\overline{s}'}) \qquad A^i_{\overline{s}'}(\omega_2^i) = \omega_1^i \qquad i = 1, ..., N$$

where ω_1^i constitutes the $\overline{s}'$-length prefix of ω_2^i; $i = 1, ..., N$.

Now we can define an operator SO of sprouting a new branch from the given one.

Definition 2.2

Let $\widehat{\omega}_{(j)}$ be the best adapted individual in some metaepoch selected from the population represented by p_{s_j} evolving in a branch of the degree j and let $s_{j+1} > s_j$. An operator SO, given by the formula $SO(p_{s_j}) = (p_{s_j}, p_{s_{j+1}})$, defines the process of construction of a population represented by the vector $p_{s_{j+1}}$, which is the initial population for the new branch of the degree $j+1$. This population is a multisubset of $\Omega_{s_{j+1}}$ and individuals for this set are selected according to the following rules:

- *$A_{s_j}(\omega) = \widehat{\omega}_{(j)}$, $\forall \omega \in \Omega_{s_{j+1}}$ (i.e. $|\omega| = s_{j+1}$)*
- *if $D \subset \mathbb{R}$, then a $(s_{j+1} - s_j)$-length suffix is selected according to the uniform probability distribution over $\Omega_{(s_{j+1}-s_j)}$*
- *if $D \subset \mathbb{R}^N$, $N > 1$, $\mu(D) > 0$, $\omega = (\omega^1, \dots, \omega^N)$; $\omega^i \in \Omega_{\overline{s}_{j+1}}$ and $N\overline{s}_j = s_j$ $(j = 1, ..., m)$, then a $(\overline{s}_{j+1} - \overline{s}_j)$-length suffix of ω^i is selected according to the uniform probability distribution over $\Omega_{(\overline{s}_{j+1}-\overline{s}_j)}$ independently for $i = 1, 2, ..., N$*

The sprouting operator can be activated or not, depending on the outcome of the prefix comparison operator defined below.

Definition 2.3

The operator $PC\colon Q \to \{0,1\}$ given by the formula:

$$PC(X, Y, s') = \begin{cases} 1 & \exists \omega_1 \in X \textit{ and } \exists \omega_2 \in Y \textit{ such that } A_s(\omega_1) = A_s(\omega_2) \\ 0 & \textit{otherwise} \end{cases}$$

where

$$\begin{aligned} Q &= \{(X, Y, s') : \ X, Y \textit{ – are multisets in } \Omega_{s''}, \Omega_{s'''}, \textit{ respectively,} \\ & \qquad s' \in \mathbb{N}, \ s' \leqslant s'', \ s' \leqslant s'''\} \end{aligned}$$

is called the prefix comparison operator.

We usually stop the algorithm when there are no new branches sprouted from the root and the evolution process in every branch is stopped. Detailed description of the HGS may be found in Kołodziej (2003a,b), Schaefer and Kołodziej (2003).

2.3. Theoretical models of the HGS

To obtain deeper understanding of how genetic algorithms work, we can construct their theoretical model. If we imagine that a population is a point in the space of all possible populations, then the effect of one generation of the genetic algorithm is to move that population to another point. As the generations go by, we obtain the trajectory of population mapped out in the space. In this way we can consider the action of the genetic algorithm to be a discrete dynamical system. Constructing the theoretical model of EA, we wish to study the properties of the trajectories through the population space that different evolutionary operators (i.e. mutation, crossover and selection) produce. There are not many theoretical models for genetic algorithms. Vose et al. (see e.g. Vose, 1999) have investigated an exact mathematical model for SGA, which forms a Markov chain in the case of finite sizes of evaluating populations. We apply this theory for construction and analysis of simple mathematical model for HGS.

According to the Vose theory (see Vose, 1999; Nix and Vose, 1992), in the case of finite sizes of populations and positive mutation rate, the SGA forms an ergodic Markov chain with states in X_n^r. When the sizes of populations are infinite, then the SGA is modelled as a sequence of trajectories of G which converges to its fixed points, if they exist.

In binary implementation of the HGS, where SGA plays the role of the mechanism of evolution, every branch in the HGS can be modelled as a Markov chain according to the Vose theory. After running several metaepochs, we have a finite family of Markov chains C_{ij}^t, where j is the branch degree, t is the local time measured in genetic epochs starting from the branch creation and $i = (i_1, ..., i_m)$, $(i_p = 0$ for $p > j)$ is the unambiguous branch identifier, which describes the "history of creation" of every branch (see Kołodziej, 2003a). In this family we have $b = \sharp\{C_{im}^t\}$ branches of the maximum degree m. That means that there are b copies of Markov chains with states in the set $X_{n_m}^{r_m} \subset \Lambda^{r_m - 1}$, (n_m denotes the size of populations evolving in these branches, s_m – length of genotypes of individuals sampled in these populations and $r_m = 2^{s_m}$).

We use this simple model to examine some asymptotic properties of the HGS. We applied Vose asymptotic theorems for the SGA (see Nix and

Vose, 1992; Vose, 1999) and SGA genetic sampling measures (see Schaefer *et al.*, 2000). We proved that the HGS does not miss any local extrema of the objective function with respect to the single population SGA, and it is more effective than the SGA in finding the multiple local optima of this function which is expressed by the ratio between the number of individuals of the single population SGA and HGS that finds the same set of local extrema. But this model has also many disadvantages, the same as the original Vose model. We could, for example, compare the properties of the HGS only with SGA.

Another idea consists in using the stochastic grammars to construct theoretical models of genetic algorithms. A big advantage of the grammar approach is that the rules of the grammar can be used to generate strings which then automatically belong to the language of the grammar. We used a context-sensitive stochastic Lindenmayer system to define another model of the Hierarchical Genetic Strategy (HGS) (see Kołodziej, 2003c). Lindenmayer systems were previously applied in genetic programming (see Koza, 1992). This idea is new and it is hard to discuss its advantages and drawbacks.

2.4. The identification of the CMM Parametric Errors by the HGS

Although the HGS is recommended mainly as a good strategy for solving multimodal problems, we reported below the solution to a problem coming from mechanical engineering. The main difficulty comes here from the objective irregularity that excludes the utilization of convex optimization methods and the need of very high accuracy. Some procedures for the evaluation of uncertainty in a coordinate measurement require the knowledge of geometrical (translational and rotational) errors of the Coordinate Measuring Machine (CMM). The traditional procedures of identification of these errors are time consuming and require special expensive equipment (see Kuntzmann *et al.*, 1990). In Kołodziej *et al.* (2004) we applied for the first time the HGS as a genetic method to deal with this problem. In this paper, we performed similar experiments for different values of parameters for the HGS as in Kołodziej *et al.* (2004).

The designation of geometrical errors of the CMM consists of three components: designation of the CMM axis concerning the reason of an error, the kind of an error (p – position, t – translation, r – rotation) and the designation of the axis on which the error has influence (refers to the errors of the kind t and p) or the designation of the rotation axis (refers to the error r). The complete model of CMM parametric errors has 21 components: positional, translational and rotational parametric (together 18 errors) and three errors of perpendicularity. The error of indication E for the CMM can be calculated as the product of the vector containing 21 components of geometrical parametric errors and the matrix M which presents the influence (weights) of each error onto the x, y and z components of the error of indication of the CMM

$$\boldsymbol{E} = \boldsymbol{K}\mathsf{M} \tag{2.3}$$

where $\boldsymbol{E} = [E_x, E_y, E_z]$ is the vector containing three components of the error of indication in a specified point in the measuring volume, and

$$\begin{aligned}\boldsymbol{K} \;\;&=\;\; [ywz, xwz, xwy, ytx, ypy, ytz, yrx, yry, yrz, xpx, xty, xtz,\\ &\qquad xrx, xry, xrz, ztx, zty, zpz, zrx, zry, zrz]\end{aligned}$$

is the vector containing 21 components of geometrical parametric errors. These parametric errors play the role of the correction coefficients of the machine. The detailed definition of the matrix M is given in Kołodziej *et al.* (2004). The knowledge of all 21 geometrical parametric errors is necessary to carry out the mathematical correction of geometrical errors and build the virtual model of CMM. In our experiments, we used the *sphere ball plate* CM Machine model (see Kuntzmann *et al.*, 1990).

The optimization problem originated from the CMM errors of identification can be defined as the problem of finding the values of geometrical errors of CMM for which the results obtained from the virtual model (simulation) differ minimally from the measurement results (in the sense of the minimum sum of squares). In fact these geometrical errors are the components of some vector $\widehat{\boldsymbol{K}}$, which play the role of optimal correction coefficients. For this problem the objective function F can be defined as the superposition of the following operators

$$F:\; \boldsymbol{K} \to \boldsymbol{E} \to \left\{ \begin{array}{c} \boldsymbol{B}^i - \boldsymbol{E} = \boldsymbol{B}^i_m \\ i = 1, ..., 25 \end{array} \right\} \to f \tag{2.4}$$

where

$\boldsymbol{K}$	–	vector containing 21 components of geometrical parametric errors
$\boldsymbol{E}$	–	vector containing three components of the error of indication in a specified point in the measuring volume
$\boldsymbol{B}^i$	–	vector of coordinates of the center of the ith ball obtained from the measurement, $B^i = [x^i, y^i, z^i]^\top$
$\boldsymbol{B}^i_m$	–	vector of measured coordinates of the center of the ith ball after correction, $\boldsymbol{B}^i_m = [x^i_m, y^i_m, z^i_m]^\top]$
(x^i_t, y^i_t, z^i_t)	–	coordinates of the center of the ith ball after transformation

X – discrete set of series of measurements of the sphere ball plate in some locations and

$$f = \sqrt{\frac{\sum\limits_{X} \sum\limits_{i=1}^{25} [(x_m^i - x_t^i)^2 + (y_m^i - y_t^i)^2 + (z_m^i - z_t^i)^2]}{25(\#X)}} \tag{2.5}$$

The value f of the objective function has an interpretation of the mean error of the measurement $[\mathrm{m}^{-3}]$ which remains after correcting the CMM accuracy by the $\boldsymbol{K}$ vector. In other words, we are looking for the vector $\widehat{\boldsymbol{K}}$ for which f is minimum.

To define the search space of the problem, we assume that the coefficients of the positional and translational parametric errors (e.g. xpx, ytz, etc.) have values in the range $(-0.003; 0.003)$ and the coefficients of the rotational parametric errors (e.g. yry) lie in the range $(-0.000004, 0.000004)$.

In our simple numerical experiments we compare the accuracy of the solution given by the HGS and the Simple Genetic Algorithm (SGA). The values of parameters for these algorithms are presented in Table 1.

Table 1. Values of parameters for SGA and HGS

Parameter	SGA	HGS		
		Level 1	Level 2	Level 3
Code length	30	10	20	30
Population size	1000	100	50	20
Mutation rate	0.015	0.03	0.015	0.01
Period of metaepoch	1	10	10	10

The results of the tests are given in Table 2.

Table 2. Results of experiments for SGA and HGS

Algorithm	Minimal value of objective	Number of fitness evaluations
SGA	0.0652 after 6000 gen.	259200
HGS	0.0028 after 1700 gen.	149200

The objective function has only one point in which the global minimum is reached. As in Kołodziej *et al.* (2004) both algorithms quickly localize their best individual close one to another: HGS after 10 generations (one metaepoch) and SGA after about 11 generations. Because we know the lower bound of the objective function (it is 0 as usual in the case of the identification problem), we can compare the algorithms analyzing the ratio of fitness values corresponding to the best individual. The HGS was about 17 times more accurate than SGA. The computational complexity of the HGS measured in evaluation of the fitness function is also considerably smaller. The obtained objective value 0.0028 is more accurate than in Kołodziej *et al.* (2004) (0.0031) and is of the same degree as declared by the CMM manufacturer. We conclude that for the smaller populations on every level of the algorithm HGS was more accurate. These results are important also for the genetic algorithms theory. Usually, genetic methods do not guarantee high accurate detection of the extremal points of the objective function. We have to start additionally a local method after genetic algorithms. The HGS gave us a very accurate solution without any additional method. The comparison with SGA showed us that the hierarchy of the processes could be the main reason of this high accuracy.

3. Optimal creative design of hierarchically represented structures

3.1. Hierarchical representation of structures

In three dimensional design problems, the solid can be defined by its partitioning into a number of smaller primitives that could not intersect (spatial partitioning). This representation is not unique. We assume that the solid is filled with the smallest possible number of primitives. Following the conclusions of Bentley (see Bentley, 1997), the Clipped Stretched Cube representation was used to represent the prototypes (the primitive is a cuboid eventually intersected by a plane defined relative to its center). Figure 2 shows examples of primitives and an object obtained out of them.

Each primitive is described by nine parameters $(x, y, z, \mathit{height}, \mathit{width}, \mathit{depth}, \alpha, \beta, d)$ and two additional attributes: *class of chair part* and *colour*. The parameters define exactly the geometry of the primitive. The point (x, y, z) describes the centre of the primitive. The clipping plane is defined by the two angels α and β and by the parameter d, which states for the distance of the plane from the centre of the cuboid. The attribute *class of chair parts* takes one value from the set {base, back, seat}. The *colour* attribute defines the colour of the primitive.

Traditionally, the meaning of the allele is defined by its position in the genotype (see Goldberg, 1989). The modified hierarchical chromosome must

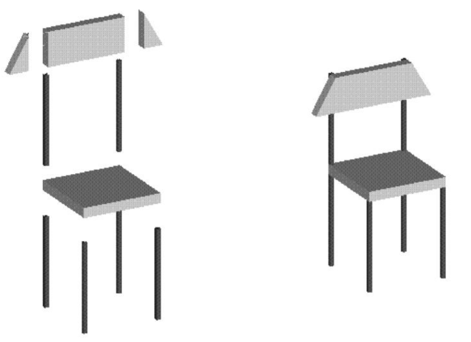
Fig. 2. Examples of primitives and an object obtained out of them

be used (see Bentley, 1997) because the number of the primitives of the graphical objects chosen to the crossover can be different (hence the length of the genotype of the individuals can be different). The meaning of the allele is thus defined by the primitive number, primitive class and the gene number. An example of the object and the hierarchical chromosome coding it are shown in Fig. 3.

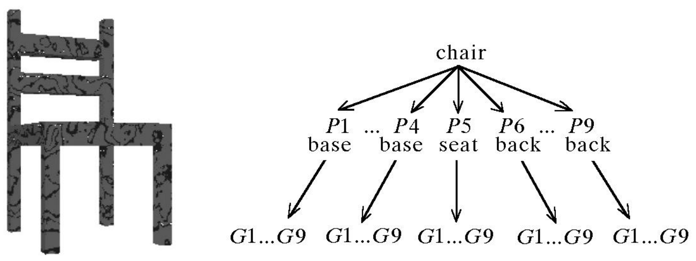

Fig. 3. An example of an object and the hierarchical chromosome coding it

In this example, the object consists of nine primitives, each of them described by nine genes. Each gene is coded as the binary string.

The representation was used to represent solids, which are solutions to real-life design problems. Thus, the assumption of the maximum number of primitives n in each chromosome could be made.

3.2. Hierarchical genetic operators

The genetic algorithm based on the modified hierarchical chromosome operates on the constant size population. It utilizes the proportional selection in order to distinct the parent for mutation and crossover. Hierarchical mutation and hierarchical crossover were used (see Paszyńska, 2002).

The hierarchical crossover, like the traditional one, is a two stage process, which consists in:

- Finding suitable crossover points (similarity points) – the triplets (P, G, B) and (P', G, B) within the parents where P is the number of the primitive in the first parent, P' in the second parent, G is the number of the gene and B is the number of the bit.
- Generating children from parents by using the crossover operator.

In the first step, primitives $P1$ and $P2$ in parents are randomly chosen. If they do not belong to the same class, the corresponding (i.e. of the same class) primitive in the second parent is searched for. If it is impossible to find such a primitive, the crossover process is terminated. After finding primitives $P1$ and $P2$ the number of the gene G is randomly chosen (Fig. 4).

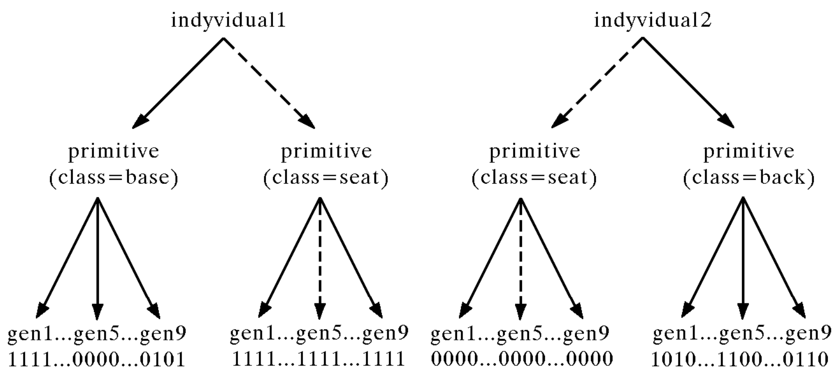

Fig. 4. Determination of similarity points between parents

In the second step, all primitives situated left from $P1$ or $P2$, respectively, and its sub-tree are copied to the children being generated – from the first parent to the first child, from the second parent to the second child (see Fig. 5).

In $P1$ and $P2$, the traditional binary crossover is used to generate two new primitives (see Fig. 6).

In the last step of the crossing process, the primitives situated right from $P1$ or $P2$ and its sub-tree are copied – from the first parent to the second child, from the second parent to the first child (see Fig. 7).

Two types of mutation are defined: the mutation of alleles and mutation of groups of alleles. The mutation operator of the first kind consists in changing one of the genes: height, width or depth. It is used with the probability 0.90. The mutation operator of the second kind is done by splitting or removing

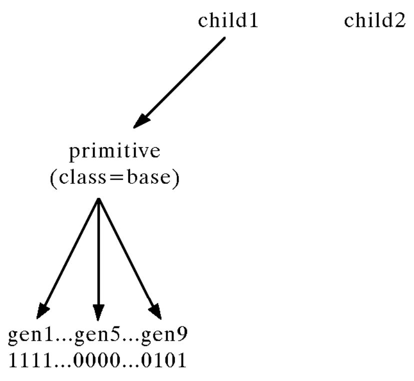

Fig. 5. The children during crossing after coping the primitives in front of a similarity point

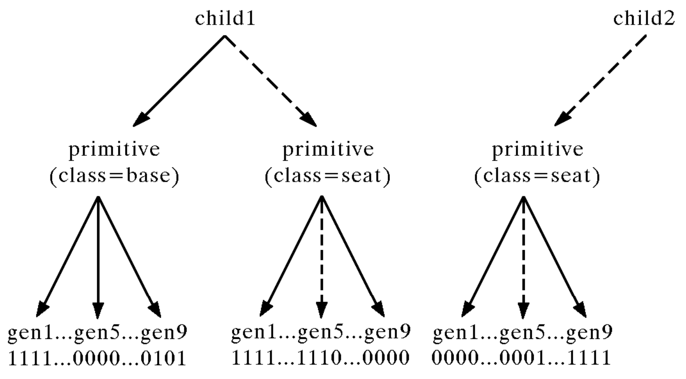

Fig. 6. The children during crossing

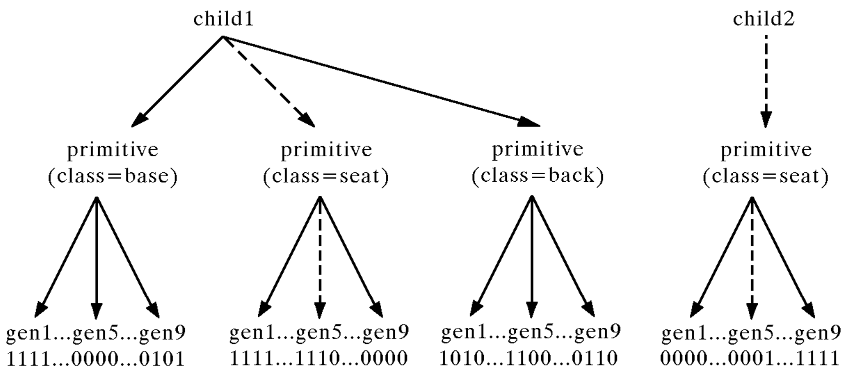

Fig. 7. The children during crossing

the group of genes constituting a single primitive. Removing a group of genes consists in removing a primitive they code from an individual. To split the promitive, the parameters describing the splitting plane (the angles α, β and the distance from the centre of the primitive d) has to be randomly chosen and the nine parameters of two newly created primitives by the splitting plane has to be calculated. The old primitive has to be removed and the two new primitives has to be added to the individual. The operator is used with the probability 0.10.

3.3. The simple test result

The object oriented application based on the genetic algorithm presented above was designed and implemented. Based on the hypothesis that the function of designs can be broken down into a number of smaller functional elements, the evaluation software is modularized into a number of evaluation modules. The modules enable the user to analyze of different designs only by picking the evaluation modules that should be used in combination. The application was designed as an open one, which means that it is possible to define new types of the evaluation.

Our case study consists in the creative designing of a chair-like structure composed of cuboids. Five modules of the evaluation were implemented:

Size: the size of the design is specified by the user-defined minimum and maximum parameters for the left, right, front, back, top and bottom of the design. Two fitness values are returned by this module: maximum size and minimum size. The module examines the six outer parameters of each phenotype. The maximum size is set to the sum of the differences between the parameters which exceed the corresponding maximum ones. The minimum size of the fitness value is calculated as the sum of the absolute values of differences between the parameters which are smaller than the corresponding minimum extents and the corresponding minimum ones.

Fragmentation: this evaluation module calculates whether the design is fragmented. If the design is fragmented, the fitness value is equal to the number of the primitives which are detached from the rest of the design. If the design is not fragmented the perfect value of zero is returned.

Center: the fitness value is set to the absolute distance between the center of the mass and the center calculated as the arithmetic sum of the vertex of all primitives.

Proportion of the chair: this evaluation module calculates whether all primitives belonging to the class *back of the seat* are well located. The location of the back is specified by the user-defined value of the back depth. The fitness value is set to the sum of the absolute differences between the user-specified value and the corresponding value calculated for all primitives belonging to the class *back of the seat*.

Flat seat surface: this evaluation module uses three user-specified values: height of the seat surface from the ground, seat width, seat depth. The fitness value is set to the sum of the absolute differences between the three user-specified values and the corresponding values calculated for the design.

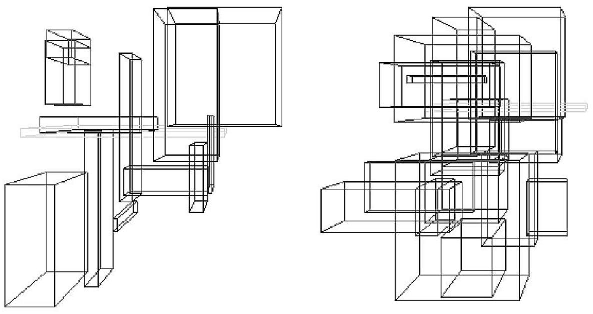
Fig. 8. Examples of randomly generated designs from the initial population

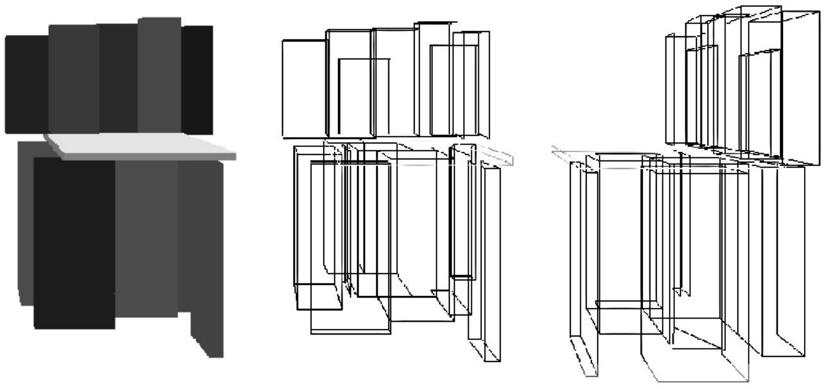
Fig. 9. The design of a chair-like structure obtained from a randomly chosen initial population after 50 steps

The applied method of the analysis of multiobjective optimisation were described by Bentley and Wakefield (1998). The multiobjective method Sum of Weighted Global Rations (SWGR) was used to calculate the overall fitness determining the position of each individual. The method based on the aggregation technique converts the fitness values for every objective into ratios, using the globally best and worst values for this objective (instead of the best and worst values in the current population, like in the SWGR). The ratios are weighted by their relative values as specified by the user, and summed up to provide a single overall fitness value for each solution.

The presented sample results were obtained from a randomly chosen population after 50 steps of the genetic algorithm operation. The population size was 120. Each individual consisted of maximum 20 cuboids randomly located in the space. The individuals from the randomly selected starting population were not the designs of the chair-like structures but constituted a randomly selected set of primitives. At the beginning they were ubderstimated by the system and in the following steps approached to more correct and more interesting designs. It can be seen that the final design fulfills all the defined requirements. For the best results, the evaluation modules like symmetry or esthetics should be defined.

4. Conclusion and perspectives

- Hierarchical genetic searches are economic (low complexity), robust and accurate with respect to classical methods for difficult optimal design problems coming from mechanical engineering and optimal creative design.
- The world is organized in a hierarchical manner and evolves continuously, so much wider applications of hierarchical genetic investigations may be expected.

References

1. Bentley P.J., 1997, Genetic evolutionary design of solid objects using a genetic algorithm, PhD Thesis, UCL London
2. Bentley P.J., Wakefield J.P., 1998, *Finding Acceptable Pareto-Optimal Solutions using Multiobjective Genetic Algorithms. Submitted to Soft Computing*, Springer Verlag Ltd. Research Report RN/98/66.

3. Bickel A.S., Bickel R.W., 1987, *Tree Structured Rules in Genetic Algorithms. Genetic Algoritmhs and Simulated Annealing*, Pittman

4. Cant-Paz E., 2000, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer

5. Goldberg D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA*, Addison-Wesley

6. Kołodziej J., 2003a, Hierarchic strategies of the genetic global optimization, PhD Thesis, Institute of Computer Science, Jagiellonian University, Cracow

7. Kołodziej J., 2003b, Modelling hierarchical genetic strategy as a family of Markov chains, In R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Waśniewski (edits.), *Parallel Processing and Applied Mathematics*, LNCS, **2328**, Springer Vlg., s. 595-599

8. Kołodziej J., 2003c, Modeling hierarchical genetic strategy as a Lindenmayer system, *Proc. of International Conference on Parallel Computing in Electrical Engineering PARELEC'2002*, Warsaw, *IEEE Computer Society*, 409-415, Los Alamitos, CA

9. Kołodziej J., Jakubiec W., Starczak M., Schaefer R., 2004, Hierarchical genetic strategy applied to the problem of the coordinate measuring machine geometrical errors, *Proc. of the IUTAM'02 Symposium on Evolutionary Methods in Mechanics*, 24-27 September 2002, Cracow, Poland, Kluver Ac. Press (accepted in print)

10. Koza J.R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press

11. Kuntzmann H., Trapet E., Waeldele F., 1990, A uniform concept for calibration, acceptance test and periodic inspection of coordinate measuring machines using reference objects, *Annals of the CIRP*, **39**, 1

12. Langdon W.B., Poli R., 2002, *Foundations of Genetic Programming*, Springer Verlag

13. Mathias K., 1991, Delta coding strategies for genetic algorithms, PhD Thesis, Department of Computer Science, Colorado State University, Fort Collins, CO

14. Nikodem P., 2000, Projektowanie za pomocą algorytmów genetycznych z wykorzystaniem teorii grafów, IIUJ, MSc Thesis (in Polish)

15. Nix A.E., Vose M.D., 1992, Modeling genetic algorithms with Markov chains, *Annals of Mathematics and Artificial Intelligence*, **5**, 1, 79-88

16. Paszyńska A., 2002, *Adapting the Hierarchical Genetic Algorithms to the Generating of the Chairs*, WAE

17. Schaefer R., Kołodziej J., 2003, Genetic search reinforced by the population hierarchy, *FOGA VII*, Morgan Kaufmann, 383-401

18. Schaefer R., Kołodziej J., Gwizdała R., Wojtusiak J., 2000, How simpletons can increase the community development – an attempt to hierarchical genetic computation, *Proc. of the 4-th Polish Conf. on Evolutionary Algorithms*, Lądek Zdrój, 2000.06.05-08, 187-199

19. Schraudolph N.N., Belew R.K., 1992, Dynamic parameter encoding for genetic algorithms, *Machine Learning*, **9**, 9-21

20. Vose M.D., 1999, *The Simple Genetic Algorithm,* MIT Press, 1999

21. Whitley D., Mathias K., Fitzhorn P., 1991, Delta coding: an iterative search strategy for genetic algorithms, In R.K.Belew and L.B.Booker (Eds.), *Proc. of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 77-84

22. Wierzba B., Semczuk A., Kołodziej J., Schaefer R., 2003, Hierarchical genetic strategy with real number encoding, *Proc. of KAEiOG'03*, Łagów Lubuski, 26-28.05.2003, 231-239

23. Wilson S., 1987, Hierarchical credit allocation in classifier system, *Proc. of the Tenth International Joint Conference on Artificial Intelligence*

Hierarchiczne obliczenia genetyczne w projektowaniu optymalnym

Streszczenie

Praca przedstawia dwa przykłady hierarchicznych, genetycznych metod optymalizacji. Sklasyfikowano dwa główne powody wprowadzenia hierarchii do modelu obliczeniowego: dla uzyskania wieloskalowego przeszukiwania z adaptowaną dokładnością oraz dla lepszego odwzorowania kształtu konstrukcji w zadaniach optymalnego projektowania kształtu. Zamieszczono rezultaty formalnej analizy proponowanych strategii oraz proste przykłady obliczeniowe.

Manuscript received April 1, 2004; accepted for print April 26, 2004