Modeling Hierarchical Genetic Strategy as a Lindenmayer System

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

Abstract. *Hierarchical Genetic Strategy is an effective tool in solving ill posed global optimization problems. We use a contex-sensitive stochastic Lindenmayer system to describe the structure of HGS. The results of simple numerical experiments are reported. We try to use this strategy in robot motion planning.*

Key words and phrases: global optimization, hierarchical genetic algorithms, hierarchical grammars, robot motion planning

1 Introduction

Hierarchical Genetic Strategy (HGS) (see [6], [7]) is an evolutionary method which gives good computation complexity in solving global optimization problems. It consists of running in parallel a set of dependent evolutionary processes. The dependency relation has a tree structure.. Simple Genetic Algorithm (SGA) (see [2]) is implemented as the basic mechanism of evolution.

The best way of determining the efficiency of genetic algorithm is to construct its theoretical model and to apply it for the analysis the properties of the algorithm.

Jacob and Rehder (see [4]) used hierarchical grammars to construct the model of neural networks. Koza (see [5]) applied L-systems in genetic programming. Using their methods we introduce a simple model based on the theory of Lindenmayer systems (see [10]), which de-scribes the structure of HGS.

We performed some numerical experiments for solving the inverse kinematic problem with six degrees of freedom in robot motion planing (see [1]). We also tried to solve the global optimization problems for chosen three multimodal functions. We compared the efficiency of HGS with other parallel and single population genetic algorithms.

2 Hierarchical Genetic Strategy

2.1 Basic objects and operations

Let $D \subset \mathbf{R}^N$ be the admissible set to global optimization problem and let $D_r \subset D$ be its finite subset. The elements of D_r are called individual phenotypes. By application of encoding one-to-one function

$$code: D_r \to \Omega_s, \tag{1}$$

Phenotypes are encoded by binary strings having length $s \in \mathbf{N}$. We denote by Ω_s the set of all genotypes of fixed length s and call it *a genetic space*. We usually identify Ω_s with the set of integers from the interval $[0,...,r-1]$, where $r = 2^s, s \geq 1$.

A collection of n elements (not necessarily different) of Ω_s is called *a population of size* n and it is represented by its frequency vector:

$$p = [p_0,...p_{r-1}]^T : \quad p_j \geq 0, \quad \sum_j p = 1, \tag{2}$$

where p_j is a proportion of element j in the population.

The main idea of HGS is running in parallel a set of dependent evolutionary processes. The processes of low order represent chaotic search with low accuracy. They only detect the promising region on the optimization landscape in which more accurate processes are activated.

The dependency relation among these processes has a tree structure. Every single process builds a branch of the tree and can be defined as a sequence of evolving populations. SGA is implemented as *the law of evolution*, which governs the progression from one generation to the next. Populations evolving in different branches may consists of individuals with different length of genotypes.

Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC'02)
0-7695-1730-7/02 $17.00 © 2002 IEEE

Definition 1. *We say that the branch has degree* $j \in \{1,...,m\}$ *if it is created by populations containing chromosomes of the length* $s_j \in \mathbf{N}$, $s_1 < s_2 < ... < s_m$.

The unique branch of the lowest degree 1 is called *root*. Populations evolving in this branch contain individuals with genotypes of the shortest length.

Let $f : \Omega_s \to \mathbf{R}^+, (f > 0)$ be a fitness function over the genetic space Ω_s. An individual $\hat{x}$ will be called *the best adapted individual* in a finite k-generation discrete evolution process

$$p^0, p^1, p^2, ..., p^k, \tag{3}$$

if it is selected as the best adapted individual for which :

$$f(\hat{x}) = \max_{x^t \in p^t, t=1,...k} \{f(x^t)\} \tag{4}$$

Definition 2. *A* k*–periodic metaepoch* $M_k (k \in \mathbf{N})$ *is a discrete evolution process which starts from the given population and terminates after at most* k *generations by selection of the best adapted individual.*

Formally, the outcome of k–periodic metaepoch started from the population may be denoted by :

$$M_k(p^m) = (p^{m+l}, \hat{x}, stop),\ l \le k, \tag{5}$$

where p^{m+l} denotes the frequency vector of resulting population , $\hat{x}$ - 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. it detects the small increment of the average fitness). The branch stop criterion flag is one, if this criterion is satisfied and zero otherwise.

The structure of HGS can be extended by sprouting of the new branches called *children* from given one. The branch from which the child is sprouted is called *a parental branch.* If the parental branch has degree $j \in \mathbf{N}$, then the degree of its child is set to $j+1$.

Let us introduce an operator A_{s_j} which "cuts out" a s_j–length prefix from the given binary string of length s_{j+1}

Definition 3. *For binary string* x *of the length* $|x| = s_{j+1}$, $A_{s_j}(x) = \tilde{x}$, *where* $|\tilde{x}| = s_j, j \in \mathbf{N}$.

We can define every string $x \in \Omega_{s_{j+1}}$ as $x = \tilde{x}\breve{\breve{x}}$, where $\tilde{x} \in \Omega_{s_j}$ is called s_j -length *prefix* and $\breve{\breve{x}} \in \Omega_{(s_{j+1}-s_j)}$ is a $(s_{j+1} - s_j)$–length *suffix* of x.

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

Definition 4. *Let* $\hat{x} \in p^{s_j}$ *be the best adapted individual in some metaepoch and let* $s_{j+1} > s_j$. *An operator* SO *given by the formula*

$$SO(p^{s_j}) = (p^{s_j}, p^{s_{j+1}}) \tag{6}$$

defines the process of construction of a population represented by the vector $p^{s_{j+1}}$, *which is an initial population for the new branch of degree* $j+1$. *This population is a multisubset of* $\Omega_{s_{j+1}}$. *It consists of individuals selected according to the following rules: for every* $x = \hat{x}\breve{\breve{x}} \in \Omega_{s_{j+1}}, (|x| = s_{j+1})$ *we have*

- $A_{s_j}(x) = \hat{x},\ \hat{x} \in \Omega_{s_j},\ |\hat{x}| = s_j$
- $\breve{\breve{x}} \in \Omega_{(s_{j+1}-s_j)},\ |\breve{\breve{x}}| = s_{j+1} - s_j$ *and* $\breve{\breve{x}}$ *is selected according to the uniform probability distribution over* $\Omega_{(s_{j+1}-s_j)}$.

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

Definition 5. *An operator* $PC : Q \to \{0,1\}$ *given by the formula:*

$$PC(X,Y,s) = \begin{cases} 1, & \exists x \in X \text{ and } \exists y \in Y : A_s(x) = A_s(y) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where

$$Q = \left\{ \begin{array}{l} (X,Y,s) : X, Y - \text{are multisets in } \Omega_z \text{ and } \Omega_r, \\ s \le z, s \le r \end{array} \right\}$$

is called a prefix comparison operator.

To start HGS we fix the following parameters:

- $1 \le s_1 \le s_2 \le ... \le s_m = s_{\max}$ - lengths of binary coded strings ($\Omega_{s_1}, \Omega_{s_2}, ..., \Omega_{s_m}$ are the gene-tic spaces associated with these strings),
- $1 \le n_1 \le n_2 \le ... \le n_m$ -sizes of populations which are multisubsets of $\Omega_{s_1}, \Omega_{s_2}, ..., \Omega_{s_m}$ respectively,
- $k \in \mathbf{N}$ - a period of metaepoch.

The evolution process starts from the initial population consisting of the shortest binary strings (s_1-length) and creates the root of the structure. The algorithm will evolve creating new branches corresponding to the populations containing individuals with different length genotypes. We denote the populations evolving in branches by:

$$p^e_{i,j}$$

where:

- e - is the global metaepoch counter
- i - the *unambiguous branch identifier* $i=(i_1,...,i_m)$, $i_p=0$ for $p>j$
- j - degree of the branch, $j\in\{1,...,m\}$. The branch contains individuals of the length s_j.

The unambiguous branch identifier $i=(i_1,...i_m)$ describes the "history of creation" of this branch. $i_1,...,i_{j-1}$ are the numbers of predecessors starting from the root. The current branch is the i_j-th consecutive child of the branch $(i_1,...i_{j-1},0,...0)$.

A process of sprouting of the new branch of degree $j+1$ form the given branch of degree j containing population $p^e_{i,j}$ is called *branch extending operation*. It is defined by branch extending procedure denoted by $BE(p^e_{i,j})$. The definition of this procedure and full description of the strategy can be found in [7].

2.2 HGS as a Lindenmayer system

Lindenmayer systems (L-systems) are usually used in theoretical biology for describing and simulating natural growth processes (see [10]). All letters in a given word are replaced in parallel and simultaneously. This feature makes L-systems especially suitable for describing fractal structures, cell divisions in multicellular organism or flowering stages of herbaceous plants.

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.

Koza (see [5]) applied Lindenmayer systems in genetic programming. Jacob and Rehder (see [4]) constructed the simple theoretical model of neural networks based on hierarchical grammar system.

Using their methods we apply a context-sensitive stochastic L-system to describe the structure of HGS.

Let us denote by $X^{n_j}_{s_j}$ the set of all possible populations of the size n_j consisting of genotypes of the length s_j, $X^{n_j}_{s_j}=(p^e_{i,j})$; $i=(i_1,...,i_m)$; $i_j\in\mathbf{N}$; $j=1,...,m$; $e=1,2,...$.

Our grammar $G(N,T,S,P)$ is characterized by:

- set of start symbols
$$S=X^{n_1}_{s_1},$$
- terminals
$$T=\{;\}\cup\bigcup_{j=1}^{m}X^{n_j}_{s_j},$$
- non-terminals
$$N=\{poplist,\ pop\}\cup\bigcup_{j=1}^{m}X^{n_j}_{s_j},$$
- production rules P specified below:

❑ $poplist := ";"\ \ pop$

❑ $pop := \left(p^e_{i,j}\middle|p^{e_1}_{i_1,j_1}\right)^*$,
$$p^e_{i,j}\in X^{n_j}_{s_j},\quad p^{e_1}_{i_1,j_1}\in X^{n_{j_1}}_{s_{j_1}}$$

❑ $p^e_{(1,0,...0),1}\xrightarrow{P_1}p^{e+1}_{(1,0,...0),1};\ \ e=1,2,...;$

❑ $p^0_{i,j}<p^1_{i,j}\xrightarrow{P_2}p^0_{\tilde{i},j+1};i=(i_1,...,i_{j-1},1,0,...,0),$
$$\tilde{i}=(\tilde{i}_1,...,\tilde{i}_{j-1},1,1,0,...,0)$$

❑ $p^e_{\hat{i},j}\xrightarrow{P_3(l)}p^0_{\hat{i},j+1};\hat{i}=(\hat{i}_1,...,\hat{i}_{j-1},\hat{i}_j,0,...,0),$
$$\hat{\hat{i}}=\left(\hat{\hat{i}}_1,...,\hat{\hat{i}}_{j-1},\hat{\hat{i}}_j,l+1,0,...,0\right)$$

The probability $P_3(l)\neq 0$ if the prefix comparison operator $PC(x,y,s_j)=0$ for all $x,y\in p^{e_1}_{\tilde{i},j+1}$ such that $\breve{i}=(\breve{i}_1,...,\breve{i}_j,l,0,...0)$, $1\le l\le e-1$, $1\le e_1\le e-1$, . $1\le j\le m-1$.

❑ $p^e_{\bar{i},j}\xrightarrow{P_4(l)}p^{e+1}_{\bar{\bar{i}},j+1};\bar{i}=(\bar{i}_1,...,\bar{i}_{j-1},\bar{i}_j,0,...,0),$
$$j=1,...,\mathrm{m}$$

The probability $P_4(l) \neq 0$ if $P_3(l) = 0$.

The following populations can be created by this grammar after 3 metaepochs depending of the outcome of prefix comparison operator applieed after the second metaepoch:

$p^3_{(1,0,\ldots,0),1}; p^2_{(1,1,0,\ldots,0),2}; p^1_{(1,2,0,\ldots,0),2}; p^1_{(1,1,1,0,\ldots,0),3}$

or

$p^3_{(1,0,\ldots,0),1}; p^2_{(1,1,0,\ldots,0),2}; p^1_{(1,1,1,0,\ldots,0),3}$

In our future research we want to apply Vose's theory of genetic algorithms (see [12]) and try to define the probability of production in this grammar.

3 Experiments

In our simple experiments we want to compare the computation complexity of HGS with SGA and other genetic algorithms specially designed for multimodal optimization.

We apply HGS to solve a global optimization problem with the Rastrigin's **(Ra)**, Schweichel's **(Sch)** and Griewangk's **(Gr)** functions defined as follows:

$$
\begin{aligned}
&\textbf{(Ra)}: f(x_1,\ldots,x_{20}) = 200 + \sum_{i=1}^{20}\left(x_i^2 - 10\cos(2\Pi x_i)\right),\\
&x_i \in [-5.12; 5.12];\\
&\textbf{(Sch)}: f(x_1,\ldots,x_{10}) = V + \sum_{i=1}^{10}\left(-x_i \sin\left(\sqrt{|x_i|}\right)\right),\\
&x_i \in [-512; 512];\\
&\textbf{(Gr)}: f(x_1,\ldots,x_{10}) = \sum_{i=1}^{10}\frac{x_i^2}{4000} - \prod_{i=1}^{10}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,\\
&x_i \in [-512; 512]
\end{aligned}
\qquad (8)
$$

All functions were converted to minimization problems.

In **Schweichel's** function V is the negative of the global minimum which is added to the function so as to move the global minimum to zero for convenience. In our experiments $V = 4189.829101$.

We compare our results with results similar experiments performed by Gordon and Whitley (see [3]) for Goldberg's SGA (see [2]), Elitist SGA (ESGA), Genitor, Island SGA (I-SGA), Island Elitist SGA (I-ESGA) and Island Genitor (I-Genitor) algorithms.

In **Elitist SGA** a copy of the best individual is always placed in the next generation.

In **Genitor** algorithm rank-based selection is applied to a sorted population. Two possible offsprings, created from two parents, are evaluated. The offspring with the higher value of fitness replaces the worst member of the population.

The classical **Island Model (I-SGA)** involves running several single population genetic algorithms in parallel. An Initial population is divided into a fixed number of subpopulations. Each "island" is a SGA with one of these subpopulations. The fixed number of individuals is periodically exchanged between the islands in process called migration. We usually use a ring topology in this migration.

In **Island Elitist SGA** and **Island Genitor** the ESGA and the Genitor algorithms are inserted into Island model in place of SGA.

In the single population algorithms (SGA, ESGA, Genitor) population size is fixed at 400 individuals and mutation rate is 0.015 .

In Island algorithms (I-SGA, I-ESGA, I-Genitor) the initial population of size 400 is divided into 10 equal subpopulations, migrations takes place after every 50 generations and mutation rate is 0.015 .

In HGS we have two levels with branches of degree 1 and 2. The values of initial parameters for HGS on every level are presented in Table 1 below:

Table 1 The values of initial parameters for HGS in experiments for (Ra), (Sch) and (Gr) functions.

Parameter	Level 1	Level 2
Code length	10	20
Population size	200	50
Mutation rate	0.03	0.015

The period of metaepoch is fixed at 100 generations.

All seven algorithms were run for 30 times for 1000 generations. Table 2 reports the number of runs in which the global optimum was found (nr) along with the average fitness of the best strings found at the end of each run (avg) for all three **(Ra)**, **(Sch)** and **(Gr)** functions :

Table 2 The values of (nr) and (avg) after 1000 generations.

Function / Algorithm	(Ra)		(Sch)		(Gr)	
	(nr)	(avg)	(nr)	(avg)	(nr)	(avg)
SGA	0	6.8	0	17.4	0	0.161
ESGA	2	1.5	16	17.3	1	0.107
Genitor	0	7.9	20	13.2	3	0.053
I-SGA	0	3.8	9	6.5	7	0.050
I-ESGA	13	0.6	13	2.6	3	0.066
I-Genitor	23	0.2	24	0.9	6	0.035
HGS	17	1.07	21	1.8	9	0.034

All concurrent algorithms are better in finding the global optimum for all fitness function than three single algorithms – SGA, ESGA and Genitor, except ESGA for Rastrigin's function.

HGS, with SGA implemented as the basic mechanism of evolution, is much more accurate than I-SGA and I-ESGA for Schweichel's and Griewangk's functions. It is also better than I-Genitor for Griewangk's function. It gives worse solution than I-ESGA and I-Genitor for Rastrigin's function and I-Genitor for Schweichel's function.

We want to apply in our future research the Genitor algorithm as the mechanism of evolution in HGS (instead of SGA) and compare the efficiency of this hierarchical algorithm with similar island models.

4 Using HGS for robot motion planning

The classical problem and a very active field in robotic is the designing a new path planner. The most of the robot motion planners are used offline: the planner is invoked with a model of the environment, it produces a path which passes to the robot controller which, in turn, execute it. In general, the time necessary to achieve this loop is not short enough to allow the robot to move in a dynamical environment. The goal is to reduce this time. Ahuactzin et al. (see [1]) applied genetic algorithms to solve the inverse kinematic problem with six degrees of freedom.

Let us denote $\theta = (\theta_1,...,\theta_6)$ a particular configuration of the arm of the robot. The *inverse kinematic problem* can be described as minimization problem $\min_{\theta} f(\theta)$ with

$$f(\theta) = \begin{cases} \|d(\theta) - X\|; & \text{if } \theta \notin \text{C} - \text{obstacles} \\ +\infty; & \text{otherwise} \end{cases} \qquad (9)$$

where $d(\theta)$ denotes the direct kinematic function and X - the desired cartesian location for the extremity of the arm.

The search space for genetic algorithms used to solve this problem is the set of *Manhattan paths of fixed length* (see[]). A *Manhattan path of length 1* is defined as the path consisting of moving each degree of freedom 1 and is denoted by $p^1 = \{\Delta\theta_1^1,...,\Delta\theta_6^1\}$. A *Manhattan path of fixed length* $l, (l \in \mathbf{N})$ is defined as the concatenation of l Manhattan paths of the length 1:

$$p \in \mathbf{R}^{l\times 6} = \{\Delta\theta_1^1,...,\Delta\theta_6^1,\Delta\theta_1^2,...,\Delta\theta_6^l\} \qquad (10)$$

Mazer et al. (see [8]) have proposed the **Ariadne's Clew** parallel genetic algorithm to solve the inverse kinematic problem with six degrees of freedom and 10 moving obstacles. This algorithm is an example of genetic tabu search algorithm (see [11]) and is based on the combination of two local planning algorithms : **Explore** algorithm and **Search** algorithm. The **Explore** algorithm collects information about the free space with an increasingly fine resolution, while, in parallel, a **Search** algorithm opportunistically checks if the target can be reached. In performed experiments they have one population of size 768 divided into 128 equal subpopulations. The mutation rate was 0.1.

We made similar experiments applying SGA and 2-level HGS algorithms. The values of the initial parameters for these algorithms are in table 3 below:

Table 3 The values of initial parameters for SGA and HGS in inverse kinematic problem.

Parameter	SGA	HGS	
		Level 1	Level 2
Code length	50	24	48
Mutation rate	0.1	0.1	0.05
Population size	770	140	50

The period of metaepoch in HGS was 2.

We report the results of the experiments for all three algorithms in Table 4 :

Table 4 The results of solving the inverse kinematic problem for 6 degrees of freedom and 10 moving obstacles for SGA, Ariadne's Clew and HGS algorithms.

	SGA	Ariadne's Clew	HGS
Number of genetic epochs needed to find the global optimum (number of fitness evaluations)	1970 (7760000)	5 (384000)	20 (429000)

The Ariadne's clew algorithm is the best in solving this problem, but it works on 128 processors in parallel. In HGS only 11 branches were created.. We want to modify, in the future, the structure of HGS by selecting the set of the best adapted individuals after every metaepoch and apply to such experiments 3 and 4-level HGS.

5 Conclusions

- HGS constitutes a kind of parallel genetic search. As opposed to Island Model the parallel processes running in HGS start dependently. Processes of higher order are activated when the promising region of existance of local extrema is detected by the lower order ones. The processes of lower order can be active until the branch stop criterion or global stop criterion is satisfied.
- We define HGS as a Lindenmayer system, but this simple model should be modified in our future work. We applied contex-sensitive stochastic grammar only to describe the structure of HGS. The probabilities of productions should be more precisely defined by application of Vose's theory of genetic algorithms (see [12]).
- The results of experiments performed for three continuous multimodal functions show that HGS can be more effective than single population genetic algorithms as well as the Island parallel genetic algorithm with SGA "engine" in finding multiple isolated extrema.
- We want to modify the structure of HGS and apply it in robot motion to get a better test results. In the performed experiments it gives worse results than specialized tabu search algorithm (Ariadne's Clew). HGS worked in this test only on 11 processors, while Ariadne's Clew algorithm – on 128.
- We suppose that the speed-up coming from the HGS parallel implementation will be also more advantageous because of nice coarse-grained parallelism.

6. References

[1] Ahuactzin J.M., Talbi G., Bessiere P., Mazer E. : "Using Genetic Algorithm for Robot Motion Planning, *ECAI'92*, Wien 1992,

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

[3] Gordon V.S., Whitley D. : "Serial and Parallel Genetic Algorithms as Function Optimizers", *ICGA'93*, S. Forest ed. Morgan-Kaufmann, 1993,

[4] Jacob C., Rehder J. : "Evolution of Neutral Net Architectures by a Hierarchical Grammar-Based Genetic system", *ICANNGA'93*, Insbruck , Springer Vlg., Wien, 1993, pp. 602-606,

[5] Koza J.R. : "Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules in Cellular automata Via Genetic Programming", *Symposium on Pattern Formation*, 1993, Claremont, CA,

[6] Kołodziej J. : " Hierarchical Genetic Strategy as a New Method in Parallel Evolutionary Computation", *Formal Methods and Intelligent Techniques in Control, Decision Making, Multimedia and Robotics*, Polish-Japanese Institute of Information Technology Press, Warsaw, October 2000, pp. 50-58,

[7] Kołodziej J., Gwizdała R, Wojtusiak J.: "Hierarchical Genetic Strategy as a Method of Improving Search Efficiency", *Advances in Multi-Agent Systems*, R. Schaefer and R. Sędziwy eds., UJ Press, Cracow 2001, Chapter 9, pp.149-161,

[8] Mazer E., Talbi G., Ahuactzin J., Bessiere P. : " The Ariadne's Clew Algorithm", *SAB'92*, Honolulu 1992,

[9] Mazer E., Ahuactzin J., Talbi E. Bessiere P., Chatroux T. : "Parallel Motion Planning with the Ariadne's Clew Algorithm", *3rd Int. Conf. Experimental Robotics ISER'93*, Kyoto, Japan, 1993,

[10] Prusinkiewicz P., Lindenmayer A. : *The Algorithmic Beauty of Plants*, Springer Vlg., New York, 1990,

[11] Talbi E., Hafidi Z., Geib J. : "Parallel Adaptive Tabu Search for Large Optimization Problems", *Parallel Computing*, vol. 24, No 14, 1998, pp. 2003-2019,

[12] Vose M.D. : *The Simple Genetic Algorithm*, MIT Press, 1990.