Managing of Cooperative Genetic Algorithms by Intelligent Agent

Halina Kwasnicka, Magdalena Gierusz
Institute of Applied Informatics, Wroclaw University of Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
halina.kwasnicka@pwr.wroc.pl, magdalena@jaworzyna.net

Abstract

Genetic Algorithms (GAs) are very popular optimization tool, although efficient applications of GAs requires users have problem with setting their parameters and used genetic operators to obtain satisfactory solution in acceptable time. We propose an intelligent agent as a control mechanism for a group of cooperating genetic algorithms. A core of the system is a family of cooperating genetic algorithms. Manager, a kind of Fuzzy Expert System, is responsible for control of GAs to assure effectiveness of the search process. It requires providin knowledge concerning influence of some parameters of GAs on tempo and mode of evolution. Such knowledge in the form of fuzzy inference rules should be included into Manager. Analyser – the third part of the system, will gather knowledge concerning progress and actual state of a working system. This knowledge will be used in the form of facts to fire suitable rules. The simulation study of efficiency of developed system is presented and discussed.

1. Introduction

Despite wide range of applications genetic algorithms are difficult to exploit especially for users who do not have detailed knowledge of them. Working with GA force user to define number of parameters (population size, coding schema, selection method, genetic operators). For selected genetic operators proper values of parameters should be specified. They determine if satisfactory solution is reachable and if it could be found in acceptable time.

De Jong was the first who has made an considerable attempt to draw up optimal set of static parameters of a GA used for function optimization. On the basis of several experiments and Holland's schema theorem [4] de Jong has worked out set of 'optimal' parameters for five test functions. This effect has initiated several research progress of GAs. It has soon turned out that 'optimal' sets of parameters give good results only for certain (usually narrow) class of problems. Therefore a number of research on genetic parameters adaptation have appeared, they can be grouped into three main streams: autoadaptation (selfadaptation), centralized methods and experiments combining both of these approaches.

Autoadaptation: it consists in encoding genetic parameters in chromosome and evolving them simultaneously with problem variables.

Chromosomes of individuals can code mutation probability [2], crossover probability or both of them [15], and a type of crossover [14]. In [12] each individual is defined as a cell with two chromosomes. First contains binary coded problem variables, the second – instructions used by a given cell to reproduce.

Central controlling: in this group we can find two main approaches: meta-level GAs and GAs based on a base of rules. Using meta-level GA was proposed by Weinberg [4]. According to Weinberg's idea the non-adaptive GA (meta-level GA) has to adjust parameters of the adaptive GA (lower-level GA). Parameters received as result of meta-level GA evolution are passed down, where lower-level GA uses them to generate and evaluate populations of constants used to simulate cells of bacteria.

Performance of GAs controlled with use of rules stored in the knowledge base makes it possible to gain and exploit experts knowledge of GA parameters adaptation. In this approach, beside classical rules [6], fuzzy logic proposed by Zadeh is frequently applied.

Most of known models work on the basis of knowledge base built with use of experts' experience and intuition [5, 10, 17]. Attempts of automatic generation of rule bases also exists, especially in cases when expert's assessment is inaccessible [7, 5].

Mixed methods: central aspect of controlling performance of GAs can be perceived (to some extent) in parameters adaptation by competition of populations. In this approach a GA consists of several subpopulations (demes) solving the same problem. Each deme has own set of genetic parameters. Parameter sets can be static for a given deme [11] or change during evolution [13, 3]. Parallel

evolution of subpopulations (every population on separate processor) is commonly applied in order to speed up performance of algorithm [8, 13, 3].

2. Developing of 'Climber' and 'Explorer' – Simulation Study

The authors have carried out several series of experiments in order to check ability to explore and exploit search space by a GA with real coded representation. Evolution has been carried out using different multimodal and multidimensional test functions. Additional assumption was to make use of genetic operators as simple as possible. Detailed description of all experiments carried out can be found in [1].

Explorer. To check the ability to explore solutions space the authors have observed population walk trough the search space, the ability to find promising solutions and to leave local optima. On the basis of those studies it has been established that a GA beeing a good explorer has small population size (8), soft selection method (roulette wheel), frequent mutations (mutation probability $p_m = 0.8$) with wide range, and no crossover. Characteristic feature of an explorer is large population diversity during whole evolution process. The authors propose a *Gaussian mutation*. A value of gene x_i is randomly selected accordingly to normal distribution with standard deviation $\delta = cd$ and mean in x_i. Factor $c \in [0; 1]$, its value for explorer is 0.17, and d denotes a range of gene values: $d = x_i^{max} - x_i^{min}$, where x_i^{min} and x_i^{max} are minimum and maximum values of gene x_i. According to the normal distribution values located in δ from x_i are randomized with probability of 68%, values located in 2δ – 95%, 3δ – 99.7%. Low range mutations appear more often.

Climber. Here, the attention was paid on precision of founded solution, tempo of climbing, repetitiveness of obtained results and ability to avoid premature convergence. Evolution was divided into two parts: climbing and solution tuning. In the first phase a population had to climb on a hill as quick as possible. In the tuning phase the task was to find most precise solution.

On the basis of experiments itF was specified that a good climber has: a larger population size (20), harder selection (roulette wheel but with keeping best individual) and uniform mutations with local range. *Uniform mutation* on a given range is modification of random mutation operator and consists in substituting value of gene x_i with randomly chosen value (uniform distribution) from $[x_i - cd, x_i + cd]$, where factors c and d are the same as for Gaussian mutation. Mutation probability should change according to climbing level (higher at the beginning, lower in tuning phase). If the convergence factor z (a ratio of the best fitness in the current population f_{max} to the average population's fitness $\overline{f}$) is lower than 1.5, fitness of individuals is calibrated to reinforce competition between individuals of climber. Linear calibration with multiplication factor $C_{mul} = 2$ is used [4].

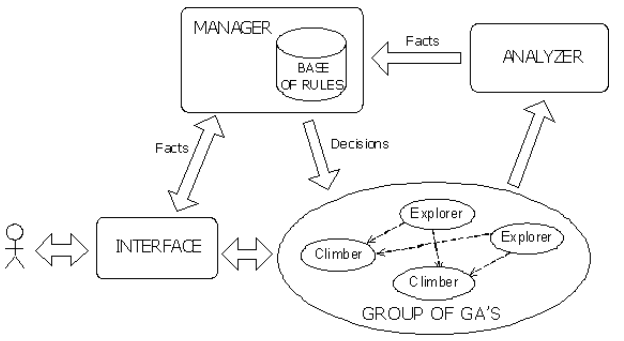

Figure 1. The architecture of *GA_AGENT*

3. Managing of a family of Cooperative Genetic Algorithms – *GA_AGENT* system

The authors have built the intelligent rule based system GA_Agent (***GeneticAlgorithmAgent***), which changes parameters of a GA dynamically and assures high level of efficiency for a given task. We propose coevolution of populations with different features. Presented system operates on the basis of knowledge base, which is internal part of the system. The knowledge is represented in the form of classical and fuzzy rules. Idea of modular system hierarchy was taken from [6] in which system IntEvol controlling binary coded genetic algorithms was presented.

Architecture of the system

GA_AGENT system consists of following modules (Fig. 1):
– *Group of cooperating GAs* with different sets of genetic parameters.
– *Analyzer* which aquisites knowledge of evolution state (by monitoring of GAs performance on the basis of several factors) and passes into *Manager* module in the form of facts.
– *Manager* – its task is to ensure maximal efficiency of solution searching. The module controls coevolution and parameters of GAs on the basis of facts (obtained from *Analyzer* and user of the system) and knowledge base (built in form of classical rules and FLC (Fuzzy Logic Controller) which tunes climber's parameters).
– *Interface* assuring communication between user and the system. It enables the user to define problem (fitness function definition and domains of its parameters) and presents optimization process and its results.

Group of genetic algorithms. In this group the two types of demes (populations of the same species) evolve: climbers and explorers. The features of these demes were

described in the previous section. Cooperation between demes consists in migrations of best individuals from explorers to climbers.

Evolution stage consists of evolving climbers and explorers simultaneously for $T = 10$ generations (experimentally established). After each generation, migrations and deletions of excessive climbers take place.

Analyzer. It is monitoring evolution process with use of following characteristics:

population average fitness:

$$\overline{f} = \sum_{i=0}^{N} f_i \tag{1}$$

where f_i – fitness value of i^{th} individual, N – population size;

population average phenotype (vector of population average genes values):

$$\overline{F} = \frac{\sum_{i=0}^{N} F_i}{N} \tag{2}$$

where F_i – phenotype of i^{th} member;

population phenotypic diversity (standard deviation of population average phenotype):

$$\sigma_F = \sqrt{\frac{\sum_{i=0}^{N} (\overline{F} - F_i)}{N}} \tag{3}$$

where $\overline{F}$ and F_i like in eq. 2;

factor of fitness growth (calculated only for climbers):

$$p = max_i \left\{ \frac{f^i_{max} - f_{min,1}}{f^{i-1}_{max} - f_{min,1}} \right\} \tag{4}$$

where i - generation's number in given evolution stage, $i = 1..T$, T – lenght of evolution stage, f^i_{max}, f^{i-1}_{max} – the best member's fitness in i-th and $i-1$ generation, $f_{min,1}$ – the worst member's fitness in first generation of first evolution stage.

distance between demes i and j (calculated for dems of the same type):

$$d^{ij} = D(\overline{F^i}, \overline{F^j}) - k \left[D(\overline{F^i}, \sigma^i_F) + D(\overline{F^j}, \sigma^j_F) \right] \tag{5}$$

where $D(\overline{F^i}, \overline{F^j})$, $D(\overline{F^i}, \sigma^i_F)$, $D(\overline{F^j}, \sigma^j_F)$ – euclidean distances, $\overline{F^i}$, $\overline{F^j}$ – average phenotypes of demes i and j, σ^i_F, σ^j_F – phenotypic diversity of demes i and j, $k = 2$ was assumed on the basis of normal distribution characteristic, according to which 95% of individuals are located in distance of two standard deviations from the population average phenotype.

If d^{ij} is negative, areas occupied by demes i and j are overlaping.

distance from explorer i *to climber* j

$$s^{ij}_{ws} = D(\overline{F^j}, F^i_{max}) \tag{6}$$

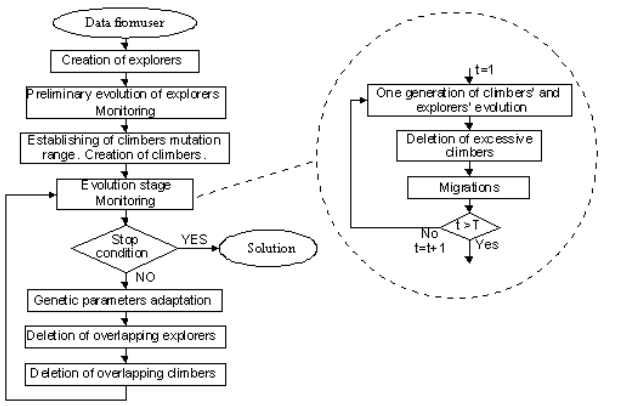

Figure 2. The architecture of *GA_AGENT*

where $\overline{F^j}$ – average phenotype of climber j in latest generation of evolution stage, F^i_{max} – phenotype of the best individual found by explorer i in the latest evolution stage.

distance from explorer i *to the closest climber*:

$$ms^i_{ws} = min_j\{s^{ij}_{ws}\} \tag{7}$$

where $j = 1..L$, L – a number of climbers.

factor describing position of the best individual found by explorer i on the slope of the hill exploited by climber j

$$s^{ij}_w = s^{ij}_{ws} - kED(\overline{F^j}, \sigma^j_F) \tag{8}$$

where $k = 3$. If s^{ij}_w factor is positive, it is assumed that the best individual in explorer i is situated beyond the hill on which climber j is climbing.

Performance of *GA_AGENT* system – *Manager*. Fig.2. shows the general scheme of the proposed system. All its steps are shortly discussed below. On the basis of optimization function's dimension and parameters' domains, *Manager* deploys explorers evenly in searching space. Initial population of explorer is situated in one point of space, all individuals are identical. It was observed [1] that creating one explorer is enough to efficient search of solutions space (for tested functions).

Preliminary evolution of explorers: To ensure enough diversity of explorers, the first 5 generations of evolution are not monitored by *Analyzer*. The next $T = 10$ generations constitute preliminary phase, during this stage of *GA_AGENT*'s performance there are no climbers.

Establishing of mutation range: Before climbers' creation *Manager* establishes mutation range for climbers according to eq. 9.

$$c = \begin{cases} 1/w & \text{if } 1/w > c_d \\ c_d & \text{otherwise} \end{cases} \tag{9}$$

where c_d – bottom limitation of mutation range equal to

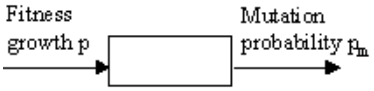

Figure 3. The scheme of FLC_1

0.025, w – an average width of range of genes values:

$$w = \frac{\sum_{i=1}^{n}(x_i^{max} - x_i^{min})}{n} \tag{10}$$

where n – an optimization function's dimension, x_i^{min} and x_i^{max} – minimum and maximum value of i^{th} function parameter (gene).

Creation of climbers: If there are more than one explorer in the search space, climbers are created according to *Migrations* procedure discussed latter on. An initial population of climber is created on the basis of the best individual X_{max} found by explorer in the latest generation. It consists of following individuals: 50% of individuals are copies of X_{max}, the other 50% are generated randomly from neighborhood of X_{max}. Value of each gene is randomized with normal distribution, a standard deviation equal to $1/5\overline{\sigma}_F$ and a mean equal to appropriate gene of X_{max}. $\overline{\sigma}_F$ is calculated as an average of σ_F (eq. 3) values calculated for explorer in monitored generations of preliminary explorers' evolution.

Described procedure is also used to perform migrations. The difference is that $\overline{\sigma}_F$ is calculated as average of σ_F values calculated for each generation of latest evolution stage.

Evolution stage – *Deletion of excessive climbers*. Excessive climber is a deme which has converged or it climbs on the earlier exploited peak. Checking existence of excessive climbers and theirs deletions are performed after each generation. It was assumed that climber i has converged if for the last $T_p = 30$ generations its best fitness had not improved (with assumed precision $\epsilon = 10^{-5}$). In that case climber's best individual is stored and the climber is deleted. If the distance from climber's i average phenotype (σ_F) to any of solutions found so far is less than three standard deviations of average phenotype of climber which found the previous solution, it is interpreted that climber i climbs on earlier exploited peak. In that case climber i is deleted. If the best fitness of climber i is worse than the stored solution, it is not stored.

Evolution stage – *Migrations*. After each generation individuals migrate from explorers to climbers. The following steps are performed for each explorer i: – Find the closest climber (eq. 7).

– If there are no climbers or for a given climber j the factor s_w^{ij} (eq. 8) is positive then create new climber (according to *Creation of climbers procedure*).

Otherwise, if $f_{max}^s > \overline{f^w}$ then replace climber's worst individual with X_{max}. X_{max} – explorer's best individual found in the latest generation, f_{max}^s – fitness of X_{max}, $\overline{f^w}$ – climber's average fitness in the latest generation.

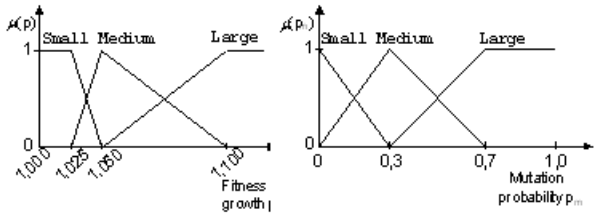

Figure 4. Fuzzy sets for FLC_1

Genetic parameters adaptation. On the basis of experiments the authors have determined good explorer's genetic parameters and operators which ensures its efficient performance, therefore *Manager* does not adapt explorers' parameters. Controlling evolution of climbers consists in performance of fuzzy logic controller FLC_1. On the basis of FLC_1 input – factor of fitness growth p (eq. 4) – a value of mutation probability p_m is being established (FLC_1 output). Scheme of FLC_1 controller is shown in Fig. 3. Fuzzy sets for controller's input and output have been defined experimentally, they are shown in Fig. 4.

FLC_1 performs on the basis of the following set of rules:

$$\begin{array}{l} \text{R1: If } p = Small \text{ then } p_m = Small \\ \text{R2: If } p = Medium \text{ then } p_m = Large \\ \text{R3: If } p = Large \text{ then } p_m = Large \end{array} \tag{11}$$

Overlapping explorers' deletion. Overlapping explorer is deme which simultaneously with other explorers is searching the same parts of solutions' space. Deletion of such demes takes place after every third evolution stage and is defined by following rule: if distance (d^{ij}, eq.5) from explorer i to explorer j is negative then one of these demes (randomly chosen) is deleted.

Overlapping climbers' deletion. it is defined similarly to the overlapping explorers' deletion. After each evolution stage the following rule is applied to each climber: if distance (d^{ij}, eq.5) from climber i to climber j is negative then delete a deme with the worse maximum fitness. It eliminates demes which exploit the same hills.

System's stop condition. Stop condition is defined by user and it is one (or few) of followings:

– a given number of generations,

– no improvement for a given number of generations with assumed precision,

– finding satisfactory solution defined by a user.

4. Efficiency of proposed Method – Simulation Study

Efficiency of the proposed system has been widely studied. Obtained results have been compared with performance of a well tuned GA with static parameters (*GA_SP*): tournament selection (tournament size equal to 2) without keeping best individual, without crossover, Gaussian mutation with $c = 0.05$ and $p_m = 0.5$, and a population size equal to 20. Experiments have been carried out for a number of different test functions with various number of hills and dimension from 2 to 20. Below we discuss shortly selected, representative results. During evolution process authors have observed population's ability to find global optima and time of performance (defined as number of calculated fitness values) needed to find global optima. For GA_Agent number of founded solutions (peaks) was additionally observed, because system's logic enables to find more than one optimum (for multimodal test functions). For each experiment at list ten, usually – twenty simulation runs have been done.

Test functions

Function Q_1. First function (F_1) is one of proposed by de Jong (5^{th} de Jong function), its dimension is equal to 2. It is multimodal with 25 hills of very similar height. Global optima is $F_1(-32, -32) = 1$. Because of evolution's nature, in experiments was used its transformed form (Q_1) defined by the eq. 12.

$$Q_1(x) = 500 - F_1(x) \tag{12}$$

Searched optimum is equal to $Q_1(-32, -32) \approx 499$

Function Q_2. It is exponential function with 3 peaks and dimension of 10. It can be analytically described with eq. 13. Section (trough the peaks) of Q_2 is shown in fig. 5.

$$Q_2(x_1, \ldots, x_{10}) = \sum_{i=1}^{l_s} h_i e^{n_i \sum_{j=1}^{10} -(x_j - x_j^i)^2} \tag{13}$$

where x_j – j^{th} function's coordinate, x_j^i – j^{th} coordinate of i^{th} peak, m – function's dimension, l_s – number of peaks, h_i – height of i^{th} peak, n_i – slope of i^{th} peak.
$h_1 = 1.5$, $(x_1^1,\ldots,x_{10}^1)$=(15,...,15), $n_1 = 0.0015$
$h_2 = 1.0$, $(x_1^2,\ldots,x_{10}^2)$=(15,...,15), $n_2 = 0.0015$
$h_3 = 2.0$, $(x_1^3,\ldots,x_{10}^3)$=(15,...,15), $n_3 = 0.0015$
Range of variables: [0; 80]. The serached optimum $Q_2(60,\ldots,60) \approx 2.0$.

Function Q_3. Q_3 (defined with eq. 14) has 2 hills and dimension of 20. Fig. 5 shows section trough the peaks of the function.

$$Q_3(x_1, \ldots, x_{20}) = \sum_{i=1}^{2} h_i e^{n_i \sum_{j=1}^{20} -(x_j - x_j^i)^2} \tag{14}$$

where: $h_1 = 1.5$, $h_2 = 1.5$, $(x_1^1, \ldots, x_{20}^1) = (15, \ldots, 15)$, $(x_1^2, \ldots, x_{20}^2) = (30, \ldots, 30)$, $n_1 = n_2 = 0,0015$.

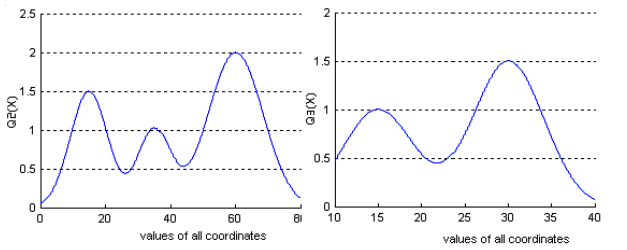

Figure 5. Section through the peaks of Q_2 and Q_3

Range of variables: [10; 40]. Searched optimum: $Q_3(30, \ldots, 30) = 1.5$.

Results of experiments

Function Q_1. The genetic algorithm with static parameters gives good results of function Q_1 optimization. Global optimum (value of 499.002) has been found in each test but time needed to find solution was strongly diversified – form 13000 fitness calculations (2^{nd} test) to 72400 (4^{th} test). Solutions finding has taken place by jumping from one hill to another one (without going down from hills).

GA_AGENT system in all tests has quickly found function's global optimum – in 90% simulation runs below 10000 fitness calculations. Obtained solutions are the same as by *GA_SP*, but the average time required to find global peak is equal to 9984 (fitness calculations) that is almost 5 times faster than the time achieved by *GA_SP* (47960 calculations). The average value of fitness calculations for *GA_AGENT* (in 1500 generations) is equal to 53156 (standard deviation is 3983). It is significantly less than performance time of *GA_SP* (75000 calculations). Optimized function has 25 hills. During 1500 generations *GA_AGENT* has found 24th of them.

Function Q_2. Population evolved by *GA_SP* in each test has got stuck in the local optimum (hill of 1.5 height). Optimization of Q_2 by *GA_AGENT* has given definitely better results. In each test the global maximum has been found (with average value of 1.999, hill's height is 2.0). Number of fitness calculations needed to find solution is on average 13679. In 8 from 10 tests system has found 2 from 3 function's hills (with height of 1.5 and 2.0). In remaining tests all three peaks has been discovered. Total evolution time of *GA_AGENT* is similar to performance time of *GA_SP* algorithm and amounts to from 60000 to 70000 depending on test's number.

Function Q_3. Population evolved with use of *GA_SP* has in each test quickly climbed on local optimum and it could not leave it during a long evolution. Population could not also tune the found local optimum. The best solution obtained during 1500 generations is equal to 0.981 (the aver-

age from 10 tests) and no improvement is observed during further evolution.
GA_AGENT has found global optimum in each test (1.5004 the average from all tests). That has taken place after on average 38128 fitness calculations, twice shorten than the evolution of *GA_SP* (75000 fitness calculations). The longest time of solution's searching amounts to 53916 fitness calculations, the shortest – 29036. In eight out of ten simulation runs the local optimum has also been found. It was furthermore observed that *GA_AGENT* is less stable, the standard deviation of fitness calculation in ten simulation runs is about 38%. Nevertheless function's optima have been found quicker than in *GA_SP* evolution.

Summarizing the results we can say that quality of founded solutions for 2-dimensional test functions is similar for both tested methods. They can find global optimum and precisely tune solution's value. *GA_AGENT* quicker discovers the global peak. It is also able to find and exploit lower hills. Number of fitness calculations during evolution (stopped after 1500 generations) is shorter for *GA_AGENT* system regardless of number of hills and range of function's parameters.

For large dimensional test functions (10 and 20) *GA_SP* algorithm converges to the local optimum and can not leave it. *GA_AGENT* gives good results of multidimensional functions optimization. It always finds the global optimum. For long enough evolution it can also find other hills of test function.

5. Summary

Simulation results show that idea of controlling of evolution performed by populations of explorers and climbers gives good results. Knowledge included in the rules' base allows for efficient controlling of evolution process. Existence of fuzzy rules lets to store knowledge which is difficult to express by numbers. Most of researches concerning real-coded GAs (which can be found in literature) focus on creation of more and more complex genetic operators. In this paper the authors have shown that there is possibility of creation of a GA on the basis of simple genetic operators. The results suggest the ability to use of GAs by people who do not possess detailed knowledge of this domain of artificial intelligence.

References

[1] Apostoluk M (Apostoluk-Gierusz) (2004) "Intelligent Agent for managing of GAs" (in Polish), University of Technology, Master Thesis, Wroclaw

[2] Back T. (1992) "Self-adaptation in genetic algorithms", http://citeseer.ist.psu.edu/14572.html

[3] Eiben A.E. (1998) Sprinkhuizen-Kuyper I.G., Thijssen B.A., "Competing crossovers in adaptive GA framework", http://citeseer.ist.psu. edu/eiben97competing.html

[4] Goldberg D.E. (1989) "Genetic Algorithms in Search, Optimization and Machine Learning"',. Addison-Wesley, 1989

[5] Herrera F., Lozano M. (1999) "Adaptive genetic algorithm based on coevolution with fuzzy behaviours", http://citeseer.ist.psu.edu/ herrera98adaptive.html

[6] Kwasnicka H., Boratyn G. (2000) "Intelligent system for managing of parameters of a family cooperating GAs" (in Polish) KAEiOG", Poland

[7] Lee M.A., Takagi H. (1993) "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques", Proceedings of 5th ICGA 1993, http://citeseer.ist.psu.edu/lee93dynamic.html

[8] Lis J. (2002) "Parallel genetic algorithm with the dynamic control parameter", 1996, (follow: [16])

[9] Michalewicz Z. (1996) "Genetic Algorithms +Data Structures = Evolution Programs", IIIed., Springer Verlag.

[10] Mulawa M., Kwasnicka H. (2000) "Efficiency imrproving by fuzzy control of GA parameters" (in Polish), KAEiOG, Poland

[11] Pham Q.T. (1995) "Competitive evolution: a natural approach to operator selection" (follow: [16])

[12] Quagliarella, D., Vicini A. (1999) "A genetic algorithm with adaptable parameters", IEEE System, Man and Cybernetics Congress, Tokyo, http://digilander.libero.it/avicini/SMC_99.pdf

[13] Schlierkamp-Voosen D., Muhlenbein H (1994) "Strategy adaptation by competing subpopulations", http://citeseer.ist.psu.edu/ schlierkamp-voosen94strategy.html

[14] Spears W.M. (1995) "Adapting crossover in evolutionary algorithms", http://citeseer.ist.psu.edu/192723.html

[15] Srinivas M., Patnaik L.M. (1994) "Adaptive probabilities of crossover and mutation in genetic algorithms", IEEE Transactions System, Man and Cybernetics

[16] Tongchim S., Chongstitvatana P., (2002) "Parallel genetic algorithm with parameter adaptation", http://citeseer.ist.psu.edu/tongchim02 parallel.html)

[17] Zeng X., Rabenasolo B (1997) "A fuzzy logic based design for adaptive genetic algorithms", 1997