The Formal Stochastic Framework for Comparison of Genetic Algorithms

P. Popela, J. Roupec, P. Ošmera and R. Matoušek
Brno University of Technology
Technicka 2, Brno, Czech Republic

Abstract - The paper purpose is to discuss the comparison of GAs. The iterations are considered as random element realizations for GAs formally defined. The quantification of algorithm capabilities inspires the use of statistical methods. The significant difference between various setups is detected by statistical tests for the test problem.

I. INTRODUCTION

A lot of different optimization algorithms involving evolutionary techniques have been developed and applied at the Brno University of Technology during the recent years for various complex technical problems in Czech industry and agriculture, see, e.g., the extreme scenario sets search for melt control problems [1] and [2], combinatorial problems in water reservoir design [3], distribution problems in the heat exchanger design [4], approximation techniques in Stirling's thermodynamic cycle optimization [5], and logical constraint transformation in irrigation pipe network reconstruction [6]. The practical experience has shown that for different problems, distinct algorithm types and setups have to be chosen. Therefore, we need to compare algorithms and tune their parameters. It is important to notice that the frequently occurring technical problems share similar important features from the algorithmic viewpoint. We will reflect features listed below in our further considerations.

(1) We are not asked to find a global optimum but we have to 'significantly improve' the known and already implemented decision. Hence, it means that we have a precise terminating criterion for the used algorithm.

(2) There is enough computational time available for the algorithm selection, comparison, and tuning period and for test computations. The results of this preprocessing phase may be successfully utilized in the algorithm implementation phase when problem data varies as the problems of the similar structure and properties are solved.

(3) In contrast, later in practice, partial real time requirements must be satisfied. Thus, it means that we have a restricted amount of time available for our computations, and so, the algorithm should be stopped after certain, in advance known, number of iterations whether the improving solution was or was not found.

(4) Because of aforementioned real time restrictions, the algorithm has to be implemented in such a way that it realizes computations as quickly as it is possible. In addition, technological demands often require that the software part of the algorithm implementation is minor relatively to its hardware part. In this case, it could be impossible to implement popular and reasonable idea of the algorithm self-learning i.e. the idea of adopting parameter changes heuristically using the experience derived from previous computations.

(5) Due to hardware facilities, the algorithm may be implemented in the parallel way.

II. OPTIMIZATION FRAMEWORK

Genetic algorithms (GAs) search the solutions of complex optimization problems. Generally, these problems may be defined as follows:

$$? = \arg\operatorname{globopt}_{\mathbf{x}} \{f(\mathbf{x}) \mid \mathbf{x} \in S\}, \tag{1}$$

where $\mathbf{x} \in S$ denotes a solution, $f(\mathbf{x})$ is an objective function, and S is a feasible set, $S \subseteq U$, where U is a universe, usually $U \subseteq \Re^n$. Then, *argglobopt* abbreviation means that a global optimum (minimum or maximum) value of the objective function f is searched on S together with the optimal argument value i.e. optimum point $\mathbf{x}_{opt}$. The sequence of symbols ?= means that we search for all global optima. In this text, symbol $\otimes \in \{\leq, \geq\}$ is used to specify whether the minimum (see $\leq$ below) or maximum (see $\geq$ below) is taken into account. Therefore, note that

$$\mathbf{x}_{opt} \in \arg\operatorname{globopt}_{\mathbf{x}} \{f(\mathbf{x}) \mid \mathbf{x} \in S\} \Leftrightarrow \forall \mathbf{x} \in S : f(\mathbf{x}_{opt}) \otimes f(\mathbf{x}). \tag{2}$$

Usually, it is enough to find just one $\mathbf{x}_{opt}$. So, instead of ?=, we write $?\in$. In addition, commonly, a modified problem is solved instead of (1). Although practitioners are frequently interested in a global minimum, they may often be fully satisfied with the 'significantly improved solution'. It might be interpreted as a local minimum with the objective function value below a certain bound F given in advance. We define:

$$? \in \arg\operatorname{locmin}_{\mathbf{x}} \{f(\mathbf{x}) \mid \mathbf{x} \in S, f(\mathbf{x}) \leq F\}. \tag{3}$$

Beyond modification (3), solution algorithms are constructed in such a way that instead of locally minimizing points specified by (3), some approximating solution points, easily findable, are searched. Denoting a set of all local minima related to (3) as $X_{\min}$, we denote a set of approximate solutions as X^*. To guarantee a tight approximation, it can be assumed that X^* satisfies the restricted distance condition. It says that the distance ρ between any approximating point and $X_{\min}$ is smaller than the given limit. It undertakes that the true minimum is not far from the approximating one i.e. $\forall v \in X^* : \inf\{\rho(u, v) \mid u \in X_{\min}\} < \varepsilon$. In some cases, these

0-7803-7282-4/02/$10.00 ©2002 IEEE

two sets may also satisfy $X_{\min} \cap X^* \neq \{\}$ or even $X_{\min} \subseteq X^*$. The approximate solution set X^* may be specified in different ways, as a set of points satisfying, e.g., the small duality gap requirement for large-scale linear programs [7] or the typical unconstrained nonlinear programming requirement saying that the gradient of the objective function should be equal to zero [8], or the integer programming requirement assigning the incumbent value [9].

The advantage of deterministic algorithms consists in the fact that under mild assumptions they often converge to the elements of approximate solution sets X^*. In this case, the algorithm is defined for a closed nonempty set U and nonempty set X^* by a point-set-mapping $A:U \to 2^U$ (see, e.g., [8]). Then, a new iteration point $\mathbf{x}[k+1]$ belongs to the set $A(\mathbf{x}[k])$ for $k \in \mathrm{N}$ where N denotes a set of natural numbers. Suppose that the sequence of $\{\mathbf{x}[k] \mid k \in \mathrm{N}\}$ is contained in a compact subset of U and there is defined a continuous descent function $\alpha: U \to \Re$ satisfying $\alpha(\mathbf{x}[k+1]) < \alpha(\mathbf{x}[k])$ for $\mathbf{x}[k] \notin X^*$. If A is a closed mapping over U-X^* (cf. [8]) then either algorithm stops in a finite number of steps with a point in X^* or it generates the infinite sequence $\{\mathbf{x}[k] \mid k \in \mathrm{N}\}$ with all accumulation points in X^* and also $\alpha(\mathbf{x}[k]) \to \alpha(\mathbf{x})$ for some $\mathbf{x} \in X^*$. For example, the descent function may be specified by the gradient norm in unconstrained convex nonlinear programming or by the objective function value in linear programming. Then, the algorithm mapping is specified, e.g., by the modified gradient direction or by the edge direction in nonlinear or linear problems, respectively. In both abovementioned cases, the algorithms converge to local minima. Under convexity assumptions (see [8]), they are also global minima.

Stochastic algorithms (see, e.g. stochastic quasigradient algorithm in [10]) are often used for global optimization nonlinear or discrete problems. In contrast to deterministic algorithms, they produce sequences that converge to X^* almost surely.

III. GENETIC ALGORITHM DESCRIPTION

Genetic algorithms commonly use heuristic and stochastic approaches. From the theoretical viewpoint, the convergence of heuristic algorithms is not guaranteed for the most of application cases, cf. II. Before we will begin to discuss GAs, we describe them (and their heuristic character) formally, cf. also discussion in [11].

At first, we assume that the result of k'th iteration is not just point $\mathbf{x}[k]$ but a subset of U denoted $X[k]$ not necessarily finite. Since now, the algorithm is described by a set-to-set mapping $A:2^U \to 2^U$, and so, $X[k+1] = A(X[k])$ similarly as before. For our heuristic ideas the situation may be more complex. At first, sets $X[k]$ may be ordered by an injection mapping $\upsilon: X[k] \to \mathrm{N}$. Hence, we denote the set of all injection mappings from all subsets of U to N as W. Then, the heuristic algorithm uses some random elements. Thus, let (Ω, Σ, P) be a probability space and related random elements be denoted by Greek letters from the end of alphabet. Then, the algorithm is described by a modified mapping $A:2^U \times W \times \Omega \to 2^U \times W$. Because set $X[k]$ depends on the random element defined on Ω, the results obtained by the repeated computations of the algorithm step randomly change. These changes may be described by the set containing random elements. To make a distinction between such set and its realization, we further denote the set of random elements as $\Gamma[k]$ and its realization remains denoted as $X[k]$. Even such a general description is possible to use for computations if sets $\Gamma[k]$, infinite by the definition, can be finitely described. However, the easiest case is when $\Gamma[k]$ is a finite set of random elements.

In comparison with common GAs, till now, we have accepted that the cardinality of $\Gamma[k]$ may vary with k. Assuming now that the cardinality of $\Gamma[k]$ is m, we may often express its realizations as $X[k] \in \Psi_m = \prod_{i=1}^{m} U$, i.e. it is an element of the Cartesian product of same universe sets U. Then, we may update the mapping A, as we do not need the injection mapping explicitly. In this case, we have $A: \Psi_m \times \Omega \to \Psi_m$. To emphasize that sets $X[k]$ and $\Gamma[k]$ are directly ordered, we denote them as matrices, so we use $\mathbf{X}[k]$ and $\mathbf{\Gamma}[k]$. We further denote a random element on Ω as ξ and its realization as ξ^s. In general, ξ may change with k. Then, $\mathbf{X}[k+1] = A(\mathbf{X}[k], \xi^s[k])$ and $\mathbf{\Gamma}[k+1] = A(\mathbf{X}[k], \xi[k])$. Thus, the behavior of the sequence of genetic algorithm (GA) iterations may be theoretically described using a discrete time multidimensional random process. Certainly, it might be questionable to identify such process by its simultaneous probability distribution function to get GA alternative complete description. However, we still may learn something about the process $\mathbf{\Gamma}[k]$ specified by GA analyzing stored time series of $\mathbf{X}[k]$. Mainly, instead of simulating GA behavior indirectly by a random process, we search for statistically based characteristics that allow us to evaluate and compare the quality of algorithms quantitatively. We have to notice that mapping A is usually composed of several subsequent mappings (see, e.g., A_S for selection operator, A_C for crossover operator already involving the population update, and A_M for mutation operator). In detail, we have three mappings: $A_S: \Psi_m \times \Omega \to \Psi_{mS}$, $A_C: \Psi_{mS} \times \Omega \to \Psi_{mC}$, and $A_M: \Psi_{mC} \times \Omega \to \Psi_m$. With the related random elements used independently in selection ξ_S, crossover ξ_C, and mutation ξ_M, we may write $\mathbf{X}[k+1] = A_M(A_C(A_S(\mathbf{X}[k], \xi_S{}^s[k]), \xi_C{}^s[k]), \xi_M{}^s[k])$ and $\mathbf{\Gamma}[k+1] = A_M(A_C(A_S(\mathbf{X}[k], \xi_S[k]), \xi_C[k]), \xi_M[k])$. Similarly, new additional operators may be easily involved. Before continuation of our discussion, we have to

mention that GAs do not solve the original problem like (1) or (3) because they deal with transformed problems of type:

$$? = \arg\operatorname{opt}_{\mathbf{x}} \{g(\pi) \mid \pi \in \Lambda, g(\pi) \le L\}, \quad (4)$$

where π denotes a population element from a huge but finite feasible set Λ obtained, e.g., by a binary coding Θ of S, and g is a fitness function related with f. Therefore, for the algorithmic purposes, we have the following related couples $\mathbf{X}[k]$ and $\mathbf{P}[k]$, $\Gamma[k]$ and $\Pi[k]$. To simplify our formulations, we do not change notation of the algorithm, its steps in Λ, and random elements with the change of its domain and we further utilize symbols A, A_S, A_C, A_M, ξ_S, ξ_C, and ξ_M. It is assumed that solving (4) using the GA, transforming it back from Λ to S by inverse relation Θ^{-1}, we get an approximate solution of (3). So, we may write: $\mathbf{P}[k+1] = A_M(A_C(A_S(\mathbf{P}[k], \xi_S^s[k]), \xi_C^s[k]), \xi_M^s[k])$ and $\prod[k+1] = A_M(A_C(A_S(\mathbf{P}[k], \xi_M[k]), \xi_S[k]), \xi_C[k])$. If K is the iteration index of the last computed iteration and $\mathbf{P}_a[K]$ is the most promising population component then we get the approximate solution of (3) as $\mathbf{x}_a = \Theta^{-1}(\mathbf{P}_a[K])$.

The abovementioned scheme unifies the description for many cases. It may be easily used for the traditional case of population with haploid individuals i.e. those with single chromosomes. However, it may work well also for the cases with more than one chromosome that are also suitable for determination of dominant genes. Dominancy based on the male/female type increases the evolutionary potential of the population, especially in varying environments, e.g., for time-dependent objective functions. In early research [12], we introduced a diploid chromosome pair mapped to a particular phenotype using a XOR dominance map, which is coded in the chromosome itself. Different approaches have been proposed in referenced literature [13]. Bagley [14], Kim [15] examined dominance as part of a genome itself, with each locus of the chromosome having a genetic value and a separate dominance value. Goldberg [16] points out crucial flaws in his method. Since mutation was not used on dominance values, they tended to converge prematurely. Brindel [17] used an extra chromosome to record dominance. These dominance vectors evolve via mutation and crossover. Collingwood's [18] and Hadad's [19] studies include using more than 2 chromosomes and a mask that specifies which of the multiple chromosomes has the dominant gene at a particular position. Hollstein [20] (later modified by Holland [21]) suggested diploidy and evolving dominance mechanism using triallelic scheme. But they tended to concentrate too much on the dominance mechanism itself and not enough on comparing the performance with haploid GAs. In [22] diploid chromosomes are created with two binary haploid chromosomes which are used to create a schema which is then used to measure the fitness. Ryan [23] uses so called shades (diploidy without dominance - additive diploidy), which can outperform all other methods in tracking the environment. Avoiding dominance means that both phenotypes have an equally likely change of expression. This may be demonstrated that standard GAs with haploid chromosomes are unable to correctly locate optimal solutions for time-dependent objective functions. This shortcoming will be simply surmounted when haploid chromosomes are substituted with diploid chromosomes [16], [24], and [25]. GAs, we use, have the following scheme [27], [28] that can also be modeled by previous formal description. Only some additional mappings are added:

1. Generation of the initial population: At the beginning the whole population is generated randomly, the members are sorted by the fitness (in descendent order). In modifications, which use a sex, two parts of the population are generated separately – males and females.

2. Mutation: The mutation is applied to each gene with the same probability, all GAs described here use $p_{mut} = 0.05$. The mutation of the gene means the inversion of one randomly selected bit in the gene.

3. Death: The classical GA uses two main operations – crossover and mutation. In GAs described in this paper we use the third operation – *death*. Every member of the population has the additional information – age. A related counter is incremented with any GA iteration. If the age of any member reaches the preset lifetime limit LT, this member 'dies' and is immediately replaced by a new randomly generated member. The age is not mutated nor crossed over.

4. Sorting by the fitness.

5. Crossover, go to step 2.

In crossover, we do not replace all members of the population. By crossover, the number of individuals corresponding to the population quarter is updated. Created individuals are sorted into the corresponding places in the population according to their fitness in such a way that the size of the population remains the same (sex of the individuals is respected). Newly created descendants of the low fitness do not have to be involved in the population. Uniform crossover is used for all genes including the sex gene (each bit of the offspring chromosome is selected separately from corresponding bits of both parents chromosomes). Four basic modifications of GA were used for testing:

A. *GA not using shades and sex (standard GA):* Haploid chromosomes are used. Every value bit is stored in one gene containing just one bit, so no redundancy occurs. To select parents for crossover the modified ranking selection strategy is used.

B. *GA using shades and sex:* The population is split into two parts – male and female (one half of initial population has sex gene preset to a 'male' value and the second one to a 'female' value). In the crossover one member of the male part and one member of the female part is chosen. Shades means, that some kind of multiploidity is used. Chromosomes contain redundant information – every value bit is stored in one three bits gene, gene values 0, 1, 2 represent value 0, gene values 5, 6, 7 represent value 1, and values 3 and 4 are 'undetermined' area – their value is set randomly (once for the whole lifetime of each gene). The structure of these three bits genes is presented on Fig. 2. The chromosome structure is shown in Fig. 1. As one can see, sex genes use shades too.

Sex genes also participate on crossover, however they are not mutated. The age is a counter used for the death operation. There are three variants of the strategy how to select parents for the crossover:

B.0 *Modified ranking selection strategy:* The selection probability of both male and female parent depends on the fitness.

B.1 *Territory defending male strategy:* The best member of the 'male' part of the population is chosen for the crossover with 80% probability, otherwise is chosen randomly. The second parent is chosen sequentially from the second ('female') part of the population.

B.2 *Pair strategy:* Parents for the crossover are chosen sequentially according to their fitness, every member of the population is used only once in one iteration.

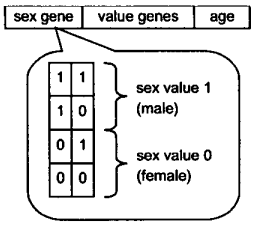

Figure 1: Structure of chromozome

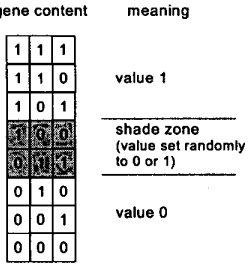

Figure 2: Structure of gene

IV. COMPARISON

In general, the following algorithm features are evaluated in optimization [8]: *generality* (required assumptions, class of solvable problems), *reliability* (robustness, achievable accuracy), *precision* (quality of iteration sequence), *sensitivity* (importance of input data changes), *preprocessing* (necessary preparation computations), *computational effort*, and *convergence*. It is known that GAs are domain independent, robust, easy of modification, and they have parallel nature. Different GAs are often compared verbally by the abovementioned list. These qualitative descriptions may be too speculative, so some objective measures should be considered. The important concept in mathematical programming is a concept of the convergence rate.

However, because of heuristic nature of GAs it cannot be used. Another idea may come from linear programming. Although the number of iterations theoretically may grow exponentially with the problem size in the simplex method, statistical evaluation of the computational experience shows that this growth is usually polynomial in practical cases. Similarly, from our description of GA by $\prod[k]$, we may use its realizations $\mathbf{P}[k]$ to derive values of g fitness function for population individuals and think about the required number of iterations on statistical basis. Then, because of the simple terminating criterion $g(\pi) \le L$ (see (4)), we may easy decide whether to stop the algorithm and count the number of realized iterations. However, this number – algorithm lifetime - is in general a random variable η. The discussion in [30] explains why the empirical probability distribution has to be used and how to determine the necessary number of iterations n for the similar solved problem instances by computing $\min\{n \in N \mid P(\eta > n) \le \alpha\}$, where α is a given probability level. The next interesting step is to set the available number of iterations (e.g., because of real time application). In this case, we have the information about the algorithm successful termination till time T specified by another random variable ξ_T. Their values are defined as follows: $\xi_T = 1 \Leftrightarrow \eta \le T$ and $\xi_T = 0 \Leftrightarrow \eta > T$. We denote distribution functions of η and ξ_T as F_η and $F_{\xi T}$ respectively. We assume that the fruitfulness of the GA is defined by the existence of probability of successful termination p. Let A and B be two different GAs with m and n runs for chosen time T, X and Y be numbers of successful terminations used for estimation of unknown probabilities of the success p_A and p_B. So, we may test whether $p_A = p_B$. Denoting $x = X/m$ and $y = Y/n$ we may use a test formula

$$U = (x-y)\left(\sqrt{z(1-z)\left(m^{-1}+n^{-1}\right)}\right)^{-1}, \qquad (5)$$

where $z = (mx+ny)(m+n)^{-1}$. Other tests could be found in [33]. When $|U| \ge u\left(\frac{\alpha}{2}\right)$, the hypothesis is rejected at the significance level α, where $u\left(\frac{\alpha}{2}\right)$ is a critical value of $N(0,1)$ distribution. See [32] for systematic presentation of comparison results by a so called 'hockey table'.

It may not be enough to consider and evaluate the algorithm behavior only by the final time percentage of the fruitfulness. Especially, we may be interested in the fruitfulness development within a time period. In this case, it might be questionable to choose between two algorithms, e.g., the first one with fruitfulness quickly increasing but remaining below a certain bound in comparison with the second one having the percentage of success increasing slower but higher. At least in the case of high number of iterations, we may statistically compare whether two different GAs are not statistically significantly different. We estimate probability distribution functions of η for two GAs empirically and then we may test their difference by the Kolmogorov-Smirnov test, see [33]. In this way, we may find that two different GA setups produce two sequences of iterations that are not different from the viewpoint of the algorithm fruitfulness.

V. TEST PROBLEM

For computational and explanatory purposes, we utilize a simple test problem although comparison ideas have also been tested for test functions, e.g., for Rastrigin function and for dynamic real world problems, e.g., fuzzy controller, see [32]. We use a simple dynamic objective function to demonstrate the efficiency and robustness of the present modification of the GA. We use a simple dynamic objective

function that simulates the behavior of previously discussed *applications* [1] - [6], e.g. changing random losses in the scenario set based two-stage melt control models or unpredictable dynamics of floods. So, we have:

$$g_1(x,t) = 1 - e^{-200(x-c(t))^2}, \quad (6)$$

determined for $x \in [2000,2090]$ and $t \in \{0,1,...,199\}$. The function $c(t)$ is determined by $c(t)=10\lceil t/20 \rceil$, where $\lceil u \rceil$ is an integer part of u, e.g. $\lceil 2.6 \rceil = 2$. Function $c(t)$ corresponds to a step function:

$$c(t) = 2000 + a(t \bmod 20) \quad (7)$$

The value of the time parameter t represents the number of iterations, so the $c(t)$ function changes after every 20th iteration.

VI. COMPUTATIONS AND RESULTS

The function (6) was used as the objective function. The sum of mean square error s in 200 iterations (every iteration corresponds to one time tic) was computed:

$$s = \sum_{i=1}^{200} \left(x_i^{best} - c(t)\right)^2 \quad (7)$$

Computations were repeated 100 times for every algorithm modification and for all values of $a \in \{5, 15, ..., 195\}$ in (7). Computations were done for different types of algorithms for lifetime limit $LT \in \{1, 2, ..., 30\}$ and for unlimited lifetime. Figures 4 and 5 show typical dependencies s on LT for different GA modifications. Three curves in graphs represent the worst, the medium and the best result obtained for every value of LT.

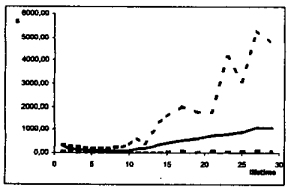

Figure 4: GA of type A (a=10)

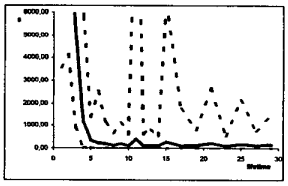

Figure 5: GA of type B1 (a=10)

Similar computations were done for testing the dependence of s on value of a in (7). Visualization of results, inspired by [31], is completed in [32].

All programs were written in the C++ programming language. The object representation of the gene, the chromosome and the GA was used.

VII. RELATED RESEARCH

Statistical approaches may also be applied to other sequences of results produced by GAs. At first, we may study how the quality of population changes. The main question is whether it permanently represents the feasible region or its transformation well. We may want to have the set that satisfies the requirement that the minimum distance is maximal. Another possibility is to satisfy the weaker requirement allowing the 'collapse' of certain individuals asking that the l'th minimum distance is maximal (so called a l-percentile criterion), see [11] and [31]. We may also analyze the complete flow of the objective function values by stream flow graphs as in [13] and [31]. All these considerations are based on the idea of taking the GA iterations as a stochastic process results and even may give suggestions of how to update some GA control parameter. At the end, we may analyze the behavior of the selected fitness function value element of population, e.g., the best one that is the optimal objective function value upper bound.

VIII. CONCLUSIONS

The paper shows how the verbal comparison of different GA versions may be supported by statistical tests confirming the difference between compared GAs by rejecting of statistical hypothesis. We have discussed a new simple diploid model of GA chromosomes, which can outperform classical haploid GAs especially when the environment is changing. At the end, we would like to discuss results for tested GAs in detail. The basic idea with the diploid models is that they dramatically increase a redundancy of coding of solutions (binary vectors). However, it was often noticed that chromosomes with diploid structure have substantially better performances then a classical GAs using haploid chromosomes (even with doubled populations). The haploid GA not using the death operator gets stuck in the first global minimum it finds, and when it is no longer global (as the time changes), this type of GA is not able to leave the old global minimum. Redundancy of coding in diploid chromosomes results in enhancing of the population diversity. The statistical results following the ideas of Section IV show that when using the proposed diploid GA for a non-stationary fitness function, the population effectively deals with the stair change of fitness landscape. The type of GA and the strategy of parent selection have the important influence to adaptability of GAs. The best results were obtained using the B.1 type of GA (territory defending male). Generally, the worst results had the A type of GA – standard haploid GA. It seems to be very interesting, that the introducing of the limited lifetime leads to improvement of adaptation capabilities of this type of GA insomuch as they are comparable to the more complicated GAs using sexual reproduction. Different types of GA have different sensitivity to the lifetime limit. Generally, we can say that the behavior of the GA using haploid chromosomes is strongly dependent on the value of the lifetime limit. The sensitivity of the GAs using diploidy chromosomes is lower. It seems, that the decreasing of the lifetime limit value leads to better result, but this problem is a little more complicated. It is necessary to mention that when the significantly low values of lifetime limit are used, the character of algorithm changes – algorithm

becomes 'pure' heuristic and the influence of crossover operator is very low. We can observe that some 'optimal' range of lifetime limit value exists. The range of suitable values of lifetime limit for haploid GAs is relatively narrow in comparison to flat and wide range of applicable LT values for GAs with sexual reproduction. We think that the best adaptability and sufficient convergence obtained by the GA version B.1 (dominant territory defending male) could be explained by the following way. It seems to be very important to split the population into two subpopulations where no exchange of genetic information exists inside these subpopulations. The first, male subpopulation contributes to fast convergence and stability of the GA and a high variety inside the female subpopulation contribute to the high adaptability of the GA.

ACKNOWLEDGEMENTS

This work has been supported by research design CEZ: J22/98: 261100009 *"Non-traditional methods for investigating complex and vague systems"*, MŠMT project MSM 260000013 and GAČR grant GA/106/01/1464.

REFERENCES

[1] P. Popela, "Heuristic search of extremal scenario sets for stochastic programs," in Proceedings of the 7th International Conference MENDEL 98, pp. 194-197, Brno, 1998

[2] P. Popela and J. Roupec, "GA-Based Scenario Set Modification in Two-Stage Melt Control Problems," in Proceedings of the 8th International Conference MENDEL 99, pp. 112-117, Brno, 1999

[3] T. Vláčil and P. Popela, "Optimum Water Reservoir Design – Flood-Related Case," in Proceedings of the 10th International Conference MENDEL 2001, Brno, 2001

[4] P. Popela, Z. Jegla, and P. Stehlík, "The Optimum Plate Heat Exchanger Design Involving Random Parameters," in Proceedings of the CHISA International Conference, Prague, 2000

[5] P. Stojan and P. Popela, "The Direct Discretisation Technique Applied to Stirling's Thermodynamic Cycle Optimisation," in Proceedings of the 9th International Conference MENDEL 2000, Brno, 2000

[6] P. Popela, S. Korsuň, P. Spitz, and T. Vláčil, "A general optimization model of the irrigation system pipe network," in Proceedings of the 9th International Conference FCE TU, Vol.10, pages 145-148, Brno, 1999

[7] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, "Linear programming and network flows," 2nd edition, John Wiley & Sons, Inc., New York, 1990

[8] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, "Nonlinear programming: theory and algorithms," 2nd edition, John Wiley & Sons, Inc., New York, 1990

[9] A. Brooke, D. Kendrick, and A. Meeraus, "Release 2.25 GAMS A User's Guide," Boyd & Fraser Publ. Co., Denvers, 1992

[10] Y. M. Ermoliev and R. J.-B. Wets, "Numerical techniques for stochastic optimization problems," Springer Verlag, Berlin, 1988

[11] P. Popela and J. Dvořák, "Global optimization and genetic algorithms," in Proceedings of the 5th International Conference MENDEL 96, pp. 205-214, Brno, 1996

[12] P. Ošmera, V. Kvasnička, J. Pospíchal, "Genetic Algorithms with Diploid Chromosomes," in Proceedings of the 6th International Conference MENDEL 97, Brno, 1997

[13] D. B. Fogel, "Evolutionary Computation," Towards a new Philosophy of Machine Intelligence, IEEE Press, 1995

[14] J. D. Bagley, "The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms," Dissertation Abstracts International 28, 1967

[15] Y. Kim, J. Kim, S. Lee, Ch. Cho, G. H. L. Kwang, "Winner Take All Strategy for Diploid Genetic Algorithm," in Proceedings of the First Asia-Pacific Conference on Simulated Evolution and Learning, 1996

[16] D. E. Goldberg, "Nonstationary Function Optimization with Dominance and Diploidy," in Proceedings of ICGA2, 1987

[17] A. Brindle, "Genetic Algorithms for Function Optimization," Unpublished doctoral dissertation, University of Alberta, Edmonton

[18] E. Collingwood, "Useful Diversity via Multiploidy," AISB Workshop on Evolutionary Computing, 1996

[19] B. S. Hadad and Ch. F. Eick, "Supporting Polyploidy in Genetic Algorithms Using Dominance Vectors," Lecture Notes in Computer Science, Evolutionary Programming VI, EP 97, USA, 1997

[20] R. B. Holstein, "Artificial Genetic Adaptation in Computer Control Systems," Doctoral Dissertation Abstracts, University of Michigan

[21] J. H. Holland, "Adaptation in Natural and Artificial Systems," Ann Arbor, The University of Michigan Press, 1975

[22] S. Lee and H. Rowlands, "A Diploid Genetic Algorithm for Finding Robust Solution in a Problem Space," in Proceedings of the 7th International Conference MENDEL 98, Brno, 1998

[23] C. Ryan, "Shades.-a Polygenic Inheritance Scheme," in Proceedings of the 6th International Conference MENDEL'97, pp. 140-147, Brno, 1997

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

[25] D. E. Goldberg and R. E. Smith, "Nonstationary function optimization using genetic algorithms with dominance and diploidy," in Proceedings of the First International Conference On Genetic Algorithms and Their Applications, Pittsburgh, PA, 24.-26.July 1985

[26] P. Ošmera, "Complex Adaptive Systems," in Proceedings of the 10th International Conference MENDEL 2001, pp.137-143, Brno, 2001

[27] J. Roupec and P. Ošmera, "Genetic algorithms with sexual reproduction for optimal fuzzy control Systems," in Proceedings of Process Control Conference 01, pp. 118, Slovak Republic, 2001

[28] J. Roupec, P. Ošmera, and R. Matoušek, "The Behavior of Genetic Algorithms in Dynamic Environment," in Proceedings of the 10th International Conference MENDEL 2001, pp.84- 90, Brno, 2001

[29] R. Matoušek and P. Ošmera, "Design of adaptive genetic algorithms based on fuzzy inference system," in Proceedigs of the 9th Fuzzy Colloquium 2001, pp.201-206, Zittau, 2001

[30] J. Roupec, P. Popela, and P. Ošmera, "The additional stopping rule for heuristic algorithms," in Proceedings of the 6th International Conference MENDEL 97, pp. 135-139, Brno, 1997

[31] R. Matoušek, P. Popela, and Z. Karpíšek, "Some possibilities of fitness-value-stream analysis," in Proceedings of the 7th International Conference MENDEL 98, pages 69-73, Brno, 1998

[32] J. Roupec, "The development of GA for optimization of fuzzy controller parameters," Ph.D. diss., Brno University of Technology, 2001

[33] J. Andel, "Statistical methods," CU, Prague, 1993