# Variable Mutation Rate at Genetic Algorithms: Introduction of Chromosome Fitness in Connection with Multi-Chromosome Representation

Matthias Kühn
Technical University Ilmenau
Ehrenbergstraße 29
98693 Ilmenau, Germany

Thomas Severin
Technical University Ilmenau
Ehrenbergstraße 29
98693 Ilmenau, Germany

Horst Salzwedel
MLDesign Technologies Inc.
2230 Saint Francis Drive
CA 94303 Palo Alto, USA

## ABSTRACT
For genetic algorithms (GAs) researchers look for optimal control parameters, such as population size or mutation rate. Early research was carried out using constant control parameters to find optimal parameter values for GA. The findings are only specific to the considered problem and therefore not suitable to be generalized. In more recent research, it was shown that the convergence rate can be increased by adaptable control parameters, e.g. mutation rate can be varied during the optimization run. Better optimization results have been achieved. It was shown how control parameters can be varied by self-adapting algorithms. The control parameters are coded within the chromosome to make them independent from the optimization problem.
In newer researches, multi-chromosome representations have been used to decompose complex problems into a number of simpler sub-problems. Each part of the problem is represented by a separate chromosome with individual representation.
Fitness values have been used to measure how good an individual fits with its environment (target criteria).
This paper investigates the effects on GA performance or the optimization results by balancing control parameters to the fitness of a chromosome (chromosome fitness).
Further it is investigated how mutation rate can be varied by chromosome fitness and whether this affects the optimization performance of the GA or the optimization results.

## Keywords
genetic algorithm, multi-chromosome, mutation rate, chromosome fitness, optimization

## 1. INTRODUCTION
In 1975 Holland published a framework on genetic algorithms[1]. Today GAs are used for optimization of diverse problems in various domains. For today's more complex problems, to better represent reality, heuristics like GAs have increased in importance.
Basic problems in using GAs are questions of genetic representation, e.g. binary/real coded, single-/multi-chromosome and the question of the optimal values for the control parameters, e.g. population size, reproduction and mutation rates.
DeJong[2], Grefenstette[3]and Schaffer et al. [4]have been focusing finding optimal control parameters for specific applications. The results cannot be generalized for other problem areas.
Fogarty[5] andHesser&Männer[6]have used variable parameter constellations by varying the control parameters during optimization run. Bäck[7]and Hinterding[8]take aware of it within the genetic representation itself (self-adapting), to get the parameters independent from the problem.
Newer researches by Juliff[9], Cavill[10]and Davidor[11]on GAs work with multi-chromosome representation to solve more complex problems. They show it is possible to decompose a complex problem into a number of simpler parts. Each part of the problem is represented by a separate chromosome andeach chromosome can use a different representation.
To measure the quality of the found solution, each individual is computed by a so called fitness function. These fitness values determine e.g. the probability that an individual is chosen for reproduction or mutation and becomes a part of the new population. By doing this, better individuals could be selected more often and could lead to a population with better individuals (solutions).
By using multi-chromosome representation it might be possible to define target criteria for each part (chromosome) of the problem. A fitness function, which rates a single chromosome within a multi-chromosome representation, is called "chromosome fitness". In case that each chromosome is rated, it is possible that one part of the problem has a high fitness value and another one is low. Keeping in mind, that parts of the problem can influence each other, it might be possible to concentrate optimization on the inferior parts and less on the good ones. This might lead to better GA performance and better optimization results.
This paper is going to analyzes how the mutation rate can be varied, as a function of chromosome fitness and its contribution to the optimization performance.
The paper first analyzes existing research in this area, describes the methodology of optimization and testing, the characteristics of GA and includes a list of the test scenarios (including the used test functions). Chromosome fitness and its use to determine mutation probability is explained. The results of the investigations are summarized.

## 2. CURRENT STAGE OF DEVELOPMENT
Researchers focused on optimal control parameter for specific problems. Usually they used common problems such as De Jong's or Schwefel's functions for test cases and for comparing their results. De Jong[2], Grefenstette[3], Schaffer et al. [4] are some of the researchers who also searched for

optimal control parameter on single-chromosome representation. In his early work De Jong [2]showed that mutation rates can have a destructive characteristic. If the mutation rate is too high, search is like a random search, regardless of other parameter settings. Based on his experiments, De Jong suggested optimal values for six control parameters, e.g. population size of 50-100 individuals, a mutation probability of 0.001 per bit (bit-flip) and single-point crossover at a rate of 0.6. His parameter set has been used in many GA implementations. Grefenstette[3]designed a secondary Meta-GA to tune the optimal control parameters for the primary GA. In his test scenario he performed five numerical test functions on six control parameters. The Meta-GA used De Jong's parameter set. Grefenstette confirmed several observations made by De Jong. Furthermore he found, that: "…the performance of GAs appears to be a nonlinear function on the control parameters" [3, p. 127]. He showed that a large generation gap as well as the elitist selection strategy (elitism) in general improved performance. Also: "In small populations (20 to 40) structures, good online performance is associated with either a high crossover rate combined with a low mutation rate or a low crossover rate combined with a high mutation rate."[3, p. 127]. Grefenstette said that mutation rate above 0.05 is in general harmful for the optimal performance of GAs[3, p. 127]. He also suggested optimal control parameters e.g. population size of 30 individuals, a mutation rate of 0.01 and for two-point crossover a rate of 0.95 (along with his test suite).

Schaffer et al. [4]performed a systematical test on control parameters affecting the performance of GAs. Therefore 10 test functions (including five functions of De Jong) were used with 6 population sizes, 10 crossover rates and 7 mutation rates. It was observed, that there is a greater sensitivity of the GA performance to mutation rate than to crossover rate. This seems to be independent from the tested functions. The optimal parameter setting was nearly the same as that of Grefenstette. For example the optimal mutation rate was seen between 0.005 and 0.01, optimal crossover rate in a range of 0.75-0.95 and a population size of 20-30 individuals [4, p. 55].

The tests produced different optimal parameter sets for different problems. Optimal control parameters seem to be dependent on each other (e.g. population size to mutation rate), on the genetic representation and on the problem to be optimized.

Usually the probability of mutation is constant throughout the optimization run. Fogarty [5] was the first to use a variable mutation rate. In his work different variants of the distribution of mutation probability were compared: (1) constant low probability over generations, (2) probability of 0.5 in initial population followed by constant low probability, (3) exponential decreasing probability, (4) constant probability across the bit representation of integers, (5) exponential increasing probability across the bit representation of integers. Summarizing he states, that varying mutation probability significantly improves performance [5, p. 108].

Hesser&Männer[6], Bäck[7]and also Bäck[12]investigated variable mutation rates. In those works mutation rates mainly depend on the genotype length and/or the population size.

Hesser&Männer[6] showed that mutation probability should be decreased during convergence, in agreement with the results of Fogarty. The benefit of variable mutation rates is seen in a high spreading of individuals in the search room on a high mutation rate at the beginning. But a high mutation rate is destructive [13, p. 56]. Therefore it is the consent that the mutation rate should be decreased during convergence [5][6].

Furthermore Bäck[7] researched on optimal mutation rates on Schwefel's and De Jong's functions. His approach showed how a GA is able to optimize the mutation rate by itself during the optimization run (self-adapting). Therefore the mutation rate is taken into the genetic representation of the individuals.

The above mentioned researches on variable or self-adapting control parameters focused on a single-chromosome representation. In this paper we focus on a multi-chromosome representation.

Pierrot&Hinterding[14] showed that mutation and crossover rates need an adaption in regard to the number of chromosomes. They furthermore indicated, that: "…a steady mutation of one variable per chromosome gives a better result than the average of one mutated variable per chromosome." [14, p. 144].

Hinterding[8] investigated on self-adapting GAs using a multi-chromosome representation. He added an extra chromosome for a numeric representation of crossover and mutation probability [8, p. 88]. It was demonstrated, that a GA that uses self-adaption, gives better results. But for easier problems, that only need a few generations, no improvements could be realized [8, p. 90]. The reason for this is that several generations are necessary to perform a good mutation rate (costs of self-adapting). But self-adapting leads to better results on complex problems and there is no need for hand-tune parameters.

## 3. METHODOLOGY OF TESTING

Mutation is the most sensitive control operator with high impact on efficient convergence of GA [4, p. 59][12, p. 7]. To investigate the effects of variable mutation rates, we define test cases on a set of test-functions (problems to be optimized). The optimization follows in a loop, as shown in Figure 1. The loop will be done as long as the wanted quality of measures is not reached.

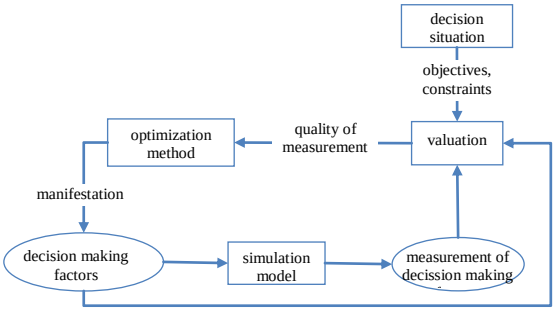


**Fig 1: Optimization of a simulation model [15, p. 4][1]**

For optimization purpose (optimization method) we use a GA with multi-chromosome representation. We considered the general sequence of action for a GA, as it is shown in Figure 2 (basic literature on GA: [1][16][17][18][19]).

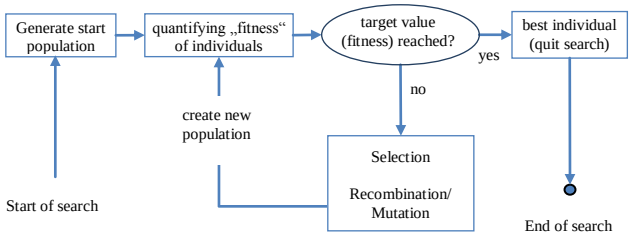


**Fig 2: General sequence of actions for our genetic algorithm (based on [20, p. 9][2])**

Following this, every individual of the population is rated by a fitness value. After selection in our approach either recombination OR mutation is carried out to create a new individual. This is necessary to assure that mutation is based on fitness value. Usually crossover is done first and upon this mutation. Thus, a new individual is created in the moment crossover is done. For mutation purpose the (before) calculated fitness value associated to the parent individual cannot be used to determine mutation rate for the new individual.

We define several test scenarios which will be simulated. For simulation purpose we use MLDesigner on an Intel i5 M430 2.27 GHz CPU running MS Windows 8.

Each test scenario is performed 100,000 times. We measure the number of generations it takes to reach a specific fitness value. For later comparison and to see whether there is an effect on performance using variable mutation rates, we calculate the mean value over all runs to compare which scenario finds the peak with the least number of generations.

## 4. CHARACTERISTICS OF GA AND TEST SCENARIOS

The initial population is set up at random and contains 20 individuals. Each individual is binary coded. As shown in figure 2 each individual is rated according to a fitness function.

After having created the initial population and according to the general sequence in Figure 2, the next step will be to select parent individuals based on their individual fitness value (selection) to perform genetic operation as mutation or crossover.

Selection is a genetic operator, which controls the search direction in the search space. As mentioned above, it determines which parents can pass their gens to a child. Individuals with better genetic material (higher fitness value) are more likely selected. During several iterations, the children population converges to an optimum. To prevent stagnation at a local optimum, it is important to keep the population variety sufficient and the genetic material diverse. Therefore parent individuals with low fitness value should also be selectable. This keeps the population “alive”. In our algorithm this is realized through normalization of fitness values of each individual according to the parent population. Based on this individuals are selected at random (“Roulette Wheel” strategy (see [17, p. 124 f.]). Thus, individuals with high fitness value are more likely selected and individuals with low fitness value can also be selected.

Crossover (recombination) is done by one-point crossover strategy (see [17, p. 128 f.]on crossover). We keep the crossover probability $p_c$ constantly at 0.6. De Jong [2] described this value as the best one in relation to his work.

Mutation rates are usually constant during the whole optimization run and so it is equal for each individual (see [17, p. 129 f.]. In our approach we use a variable mutation rate as a function of number of generations and what we call chromosome fitness. Mutation is always done by bit-flip and it happens at each bit position with probability $p_m$. The reason for this is, the little length of chromosomes in our representation in combination with low mutation probabilities like $p_m$=0.001.

Finally we use elitism, which is a selection method that forces the GA (only) to pass the best individual without any changes to the next generation (see [17, p. 124 f.] on elitism). Otherwise such an individual can be destroyed and be lost for the population.

In the following we define three test cases and two test-functions (problems) which should be optimized. As a consequence of this several test-scenarios will be performed:

Case 1 (Constant):
Multi-chromosome representation - mutation rate is constant,

Case 2 (Fogarty):
Multi-chromosome representation - mutation rate is decreasing during the optimization run in the same for all chromosomes,

Case 3 (Variable):
Multi-chromosome representation - mutation rate is variable during the optimization run for each chromosome and is based on the chromosome fitness.

As test functions we use De Jong’s (1) and Schwefel’s function (2).

De Jong’s function (DF):

$$f_{DJ}(x,y) = x^2 + y^2 \qquad (1)$$
$$-4 \le x,y \le 4$$

Schwefel’s function (SF):

$$f_S(\vec{x}) = \sum_{i=1}^{n} -x_i * \sin\sqrt{|x_i|} \qquad (2)$$
$$-500 \le x_i \le 500$$
$$n \in N;\ \vec{x^*} = (x_1^*, x_2^*, x_3^*, \dots, x_n^*)$$

For these functions the minima are known as follows (3, 4):

$$f_{DJ}(x^*, y^*) = 0 \qquad (3)$$
$$\text{at: } x^* = 0;\ y^* = 0$$

$$f_S(\vec{x^*}) = -n * 418{,}9828872 \qquad (4)$$
$$\text{at } x_i^* = 420.968746\ \forall i$$
$$n \in N;\ \vec{x^*} = (x_1^*, x_2^*, x_3^*, \dots, x_n^*)$$

For both test functions each of the variables should be represented by a separate chromosome and therefore we implement the GA as haploid multi-chromosome representation with two chromosomes. Therefore we split

each problem into two parts. Because both functions are sum functions, it can easily be done as follows:
For De Jong's function the parts are (5, 6):

$$f_{DJ_1} = x^2, \quad (5)$$

$$f_{DJ_2} = y^2. \quad (6)$$

Thus x is represented by the first chromosome and y is represented by the second chromosome.

For Schwefel's function the parts are (7, 8):

$$f_{S_1}(x_1) = -x_1 * \sin(\sqrt{|x_1|}), \quad (7)$$

$$f_{S_2}(x_2) = -x_2 * \sin(\sqrt{|x_2|}). \quad (8)$$

Thus the first chromosome represents $x_1$ and the second one $x_2$.

For De Jong's function each chromosome has 13 bits, caused by a possible input value with 3 fractional digits to enlarge the granularity of the search space. For Schwefel's function each chromosome has 17 bits, caused by a possible input value with 2 fractional digits for the same reason.
In this paper we distinguish between the fitness of an individual $F_I$ and the fitness of a chromosome $F_C$. The individual fitness determines the goodness of the individual (solution) as a whole and determines the probability that a parent individual is selected for mutation, reproduction or elitism. The chromosomal fitness is the fitness value of a part of an individual (a chromosome) and we use it to determine the variable mutation rate for this specific chromosome in case that the individual is selected for mutation.
We tested a set of fitness functions before (e.g. exponential, linear, square root) to see which is suitable best for a defined fitness level to be reached. At some fitness functions we set a limit to 2 or to 10. So the fitness function sets all values which were greater than 2 or 10 on a constant low fitness value. Best solutions and best performances were reached with an exponential and a linear fitness function (see Figure 3 on De Jong's function).

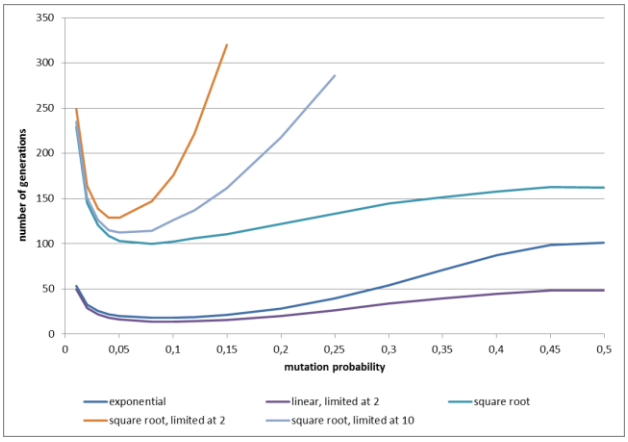


**Fig 3: Effects on GA performance based on six fitness functions on De Jong's function**

We decided to use the exponential fitness function. According to this the individual fitness functions $F_I$ are as follows (see Figure 4, Figure 5):

De Jong:

$$F_{I_{DJ}}(x,y) = e^{-f_{DJ}(x,y)} \quad (9)$$

Schwefel:

$$F_{I_S}(x_1,x_2) = e^{(-0{,}01*(f_S(x_1,x_2)+f_S(x_1{}^*,x_2{}^*)))} \quad (10)$$
$$f_S(x_1{}^*,x_2{}^*) = 837{,}9657744$$

By adding $f_S(x_1{}^*,x_2{}^*)$ to $F_{I_S}$ (equation 10), we adjust the fitness function to the range of possible values (search space). By using 0.01 we define the lowering of $F_{I_S}$ within the search space (see Figure 5).

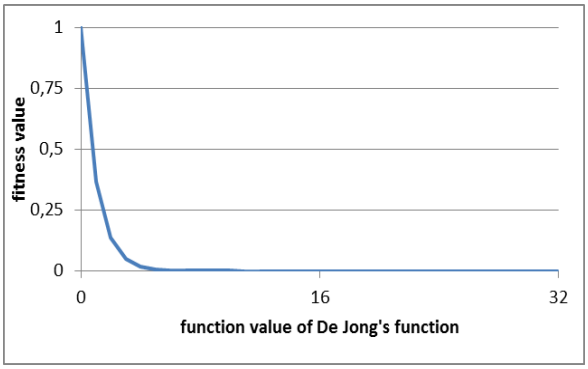


**Fig 4: Individual fitness function for De Jong's function**

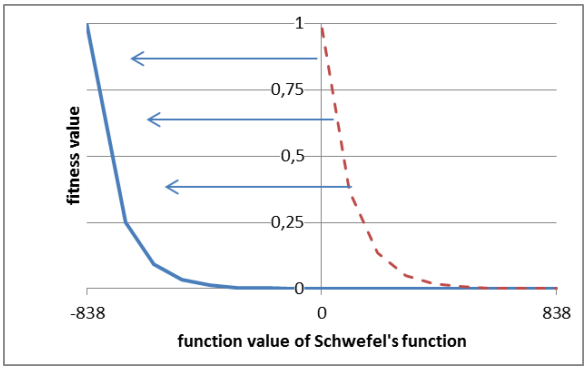


**Fig 5: Individual fitness function for Schwefel's function**

It is seen that we set a high selection pressure by the early lowering of the fitness values. Ochoa et al. said that at a high selection pressure a higher mutation rate should be used [21, p. 321]. That is why we start the test scenarios with a high mutation probability of 0.5. We assume, that mutation rate above 0.5 is too destructive to achieve good results (compare [13]).
The optimization target is an individual fitness value of 0.995 for De Jong's and Schwefel's function.
To investigate the effects, we perform scenarios only with mutation and selection as genetic operators. This is called

native evolution (NE). We also investigate the effects by using mutation, reproduction (crossover) and elitism (MCE). The parameter sets for our test cases will be as follows:
Case 1 (Constant):

In this case the mutation rate is constant during the optimization run. Therefore we use a set of seven possible values to be tested: 0.001/0.01/0.02/0.05/0.1/0.2/0.5.

Case 2 (Fogarty):

In this case the mutation rate is decreasing during the optimization run. The mutation probability for the first/initial generation is 0.5. In all following generations we calculate the mutation probability $p_m$ as Fogarty [5] did (equation 11, 12). Instead of 0.11375 we use 0.5.

Fogarty:

$$p_{m_1} = 0.5 \tag{11}$$

$$p_{m_i} = \frac{1}{number\ of\ bits\ in\ the\ population} + \frac{0{,}5}{2^{current\ \ generation\ \ number}} \qquad i = 2 \ldots n \tag{12}$$

Case 3 (Variable):

In this case the mutation rate is varying based on the chromosome fitness $F_C$. The chromosome fitness based on an exponential function is calculated as follows (13-16):

De Jong:

$$F_{C_{DJ}}(x) = e^{-2 * f_{DJ\,1}(x)} \tag{13}$$

$$F_{C_{DJ}}(y) = e^{-2 * f_{DJ\,2}(y)} \tag{14}$$

Schwefel:

$$F_{C_S}(x_1) = e^{(-0{,}02 * (f_{S_1}(x_1) + f_S(x_1^*)))} \qquad f_S(x_1^*) = 418{,}9828872 \tag{15}$$

$$F_{C_S}(x_2) = e^{(-0{,}02 * (f_{S_2}(x_2)) + f_S(x_2^*)))} \qquad f_S(x_2^*) = 418{,}9828872 \tag{16}$$

In $F_{C_{DJ}}$ (13, 14) we use the factor of 2 to equalize the lowering rate to the reduced search space of the single chromosome. In $F_{C_S}$(15, 16) we changed the factor from 0.01 to 0.02 to equalize the lowering rate and replace $f_S(x_1^*, x_2^*)$ by $f_S(x_1^*)$ resp. $f_S(x_2^*)$ to adjust the fitness function to the reduced search space.
The chromosome fitness determines the mutation rate for a specific chromosome. It is set high at low chromosome fitness values and vice versa. The mutation probability of a chromosome can vary between 0.5 and 0.001. To calculate the specific mutation rate of a chromosome either a linear or an exponential decreasing function is used (see **Error! Reference source not found.**).

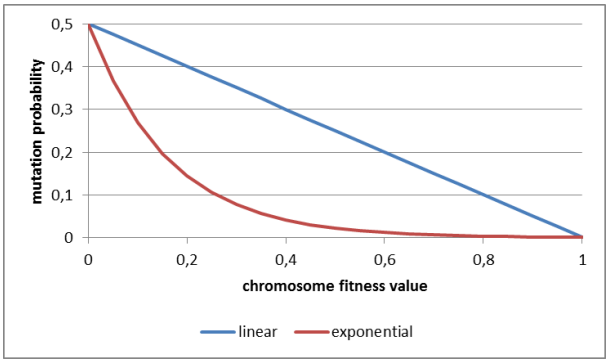


**Fig 6: Used functions for decreasing mutation probability**

The mutation rate is calculated as follows (17, 18):

linear:

$$p_{m_c} = 0.5 - 0.499 * F_C \tag{17}$$

exponential:

$$p_{m_c} = 0.5 * e^{(-6.2146 * F_C)} \tag{18}$$

As it can be seen, the mutation probability is based on the above mentioned maximum probability of 0.5 as well in the linear decreasing function as in the exponential decreasing one. To reach the minimum mutation probability of 0.001 at the maximum chromosome fitness value of 1, a scaling is done by a scaling factor of 0.499 in the linear function and a scaling factor 6.2146 in the exponential function.
For De Jong's and Schwefel's functions the global minimum will be reached, if each of its parts reaches its minimum. It may be possible that this is not always the case, especially then if the target criteria are interdependent. In these cases the optimum is seen in a balance among the target criteria and the optimal balance may lead to a low fitness value for the chromosomes and in a consequence to unnecessary high mutation rates. Therefore further investigation has to be done. A solution might be to add a decreasing factor of e.g. 0.998 to each generation to calculate the mutation rate (according to Fogarty). This can additionally reduce the mutation rate during optimization run.
The descriptions above lead to 40 test scenarios in total (see **Error! Reference source not found.** and Table 2):

**Table 1: Test scenarios on native evolution**

| Test-Function | Scenarios on GA with native evolution |
|---|---|
| De Jong's function | Constant (0.001/0.01/0.02/0.05/0.1/0.2/0.5)<br>Fogarty<br>Variable (linear)<br>Variable (exponential) |
| Schwefel's function | Constant (0.001/0.01/0.02/0.05/0.1/0.2/0.5)<br>Fogarty<br>Variable (linear)<br>Variable (exponential) |

**Table 2: Test scenarios on mutation, crossover and elitism (MCE)**

| Test-Function | Scenarios on GA with mutation, crossover and elitism (MCE): |
|---|---|
| De Jong's function | Constant (0.001/0.01/0.02/0.05/0.1/0.2/0.5)<br>Fogarty<br>Variable (linear)<br>Variable (exponential) |
| Schwefel's function | Constant (0.001/0.01/0.02/0.05/0.1/0.2/0.5)<br>Fogarty<br>Variable (linear)<br>Variable (exponential) |

These scenarios are distinguished between 4 variants based on the two test functions and the used genetic operators (DF/NE, DF/MCE, SF/NE, SF/MCE).

## 5. EXPERIMENTAL RESULTS

We calculated our test scenarios and measured the effects on GA performance. Therefore we compared the number of generations to find out which scenario needs the least number of generations to reach the given fitness level. In the following we describe our results in the test cases of De Jong's and Schwefel's function (see Figures 7 – 10):

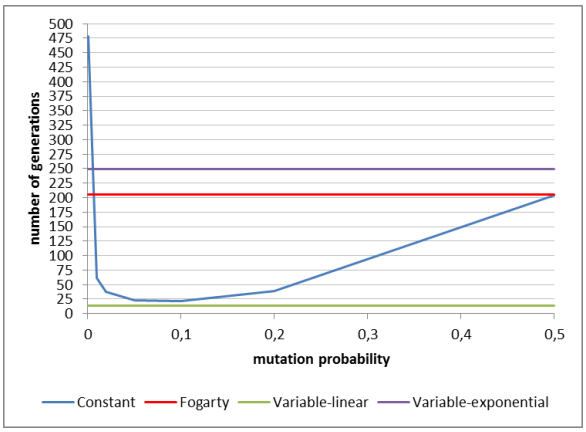


**Fig 7: Results of variant DF/NE**

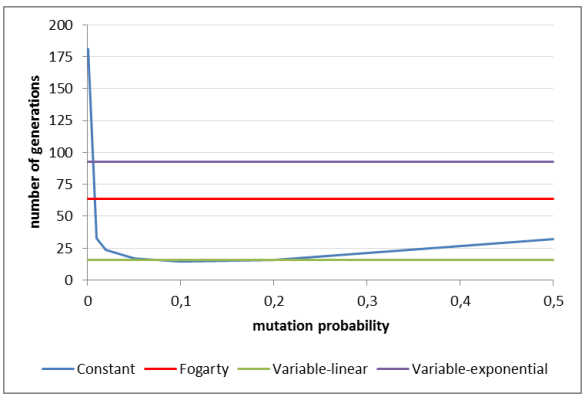


**Fig 8: Results of variant DF/MCE**

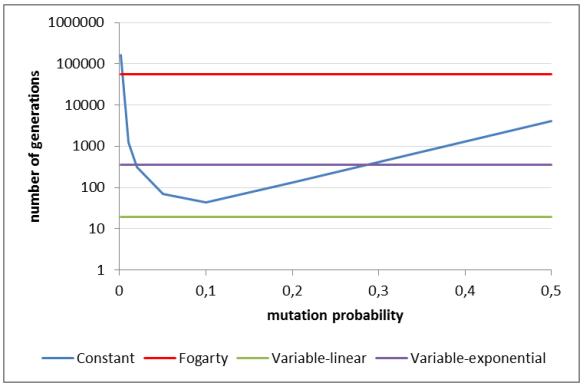


**Fig 9: Results of variant SF/NE**

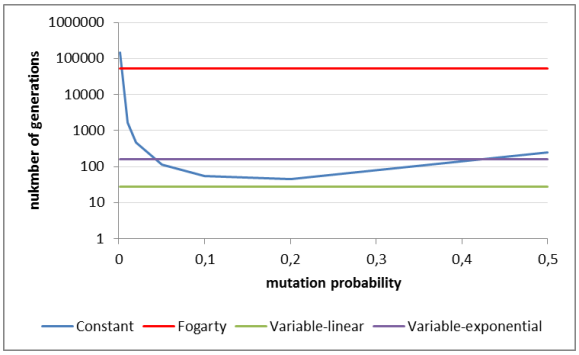


**Fig 10: Results of variant SF/MCE**

The constant mutation rates lead to a curve with a high number of generations for low and high mutation rates. The best performance for case 1 is reached at a mutation probability in a range of 0.05 and 0.2.

Within our scenarios in case 2 (Fogarty) an acceptable GA performance was reached. Most of the constant test scenarios in case 1 reached a better performance. The lowering of the population probability at the Fogarty scenarios might be too fast when it is cut in nearly a half at each generation. Thus it indicates a benefit that no hand-tune is needed. But it seems that despite of that an adjustment to the specific problems is necessary.

The scenarios in case 3 (Variable) lead to a better performance with a linear decreasing mutation rate and a worse performance with a exponential decreasing mutation rate, compared with case 2 (Fogarty). We observed this for De Jong's and Schwefel's function. It arises the question, why is the linear decreasing of the mutation rate much better than the decreasing with the exponential function. We assume that the reason therefore can be seen in a too fast lowering of the mutation probability in relation to the chromosome fitness with the exponential decreasing function (see Figure 6).

In case 3 we also observed, that in 3 of 4 variants the linear decreasing scenarios led to a better GA performance as it was reached in scenarios with constant mutation rates (see Figure 7-10 and Table 3-4).

We cannot explain the fact, that GA performance with variable mutation rate based on chromosome fitness with a linear decreasing function is so good, in comparison with constant and Fogarty scenarios. We assume that the reason for this can be found in the splitting of the problem into parts and therefore chromosome based individual mutation probabilities lead to this effect. Further research on these aspects is recommended.

**Table 3: Performance results of test scenarios on De Jong's function (mean value of 100,000 runs)**

| Test-Scenario | DF/NE | DF/MCE |
|---|---|---|
| Constant 0.001 | 478.4 | 180.9 |
| Constant 0.01 | 61.6 | 32.9 |
| Constant 0.02 | 37.4 | 23.3 |
| Constant 0.05 | 23.4 | 16.7 |
| Constant 0.1 | 22.1 | 14.5 |
| Constant 0.2 | 39.2 | 15.5 |
| Constant 0.5 | 203.4 | 31.8 |
| Fogarty (0.5 – 0.001) | 205.0 | 63.6 |
| Variable (linear 0.5 – 0.001) | **13.7** | **15.9** |
| Variable (exponential 0.5 – 0.001) | 248.9 | 92.6 |

**Table 4: Performance results of test scenarios on Schwefel's function (mean value of 100,000 runs)**

| Test-Scenario | SF/NE | SF/MCE |
|---|---|---|
| Constant 0.001 | ca.160,000 | ca.150,000 |
| Constant 0.01 | 1,246.3 | 1,635.2 |
| Constant 0.02 | 307.1 | 466.3 |
| Constant 0.05 | 69.8 | 112.4 |
| Constant 0.1 | 44.5 | 55.0 |
| Constant 0.2 | 133.0 | 45.3 |
| Constant 0.5 | 4,080.0 | 252.2 |
| Fogarty (0.5 – 0.001) | ca.57,000 | ca.53,500 |
| Variable (linear 0.5 – 0.001) | **19.1** | **27.3** |
| Variable (exponential 0.5 – 0.001) | 351.5 | 159.2 |

Figure 11 and 12 illustrate the resulting mutation rates during the optimization run in case 2 and case 3 for De Jong's and Schwefel's functions. As it can be seen, there is a constant decreasing of mutation rate in the Fogarty scenario as well as in the variable scenarios. The decreasing in Fogarty's scenarios is done by a factor in relation to the number of the generation. Related to the variable mutation rate in case 3 we assume that the decreasing is caused by convergence of the population to an optimum in the optimization run.

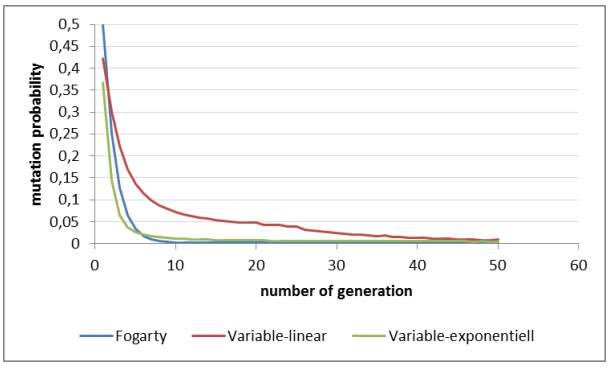


**Fig11: Resulting mutation rate during optimization run at De Jong's function**

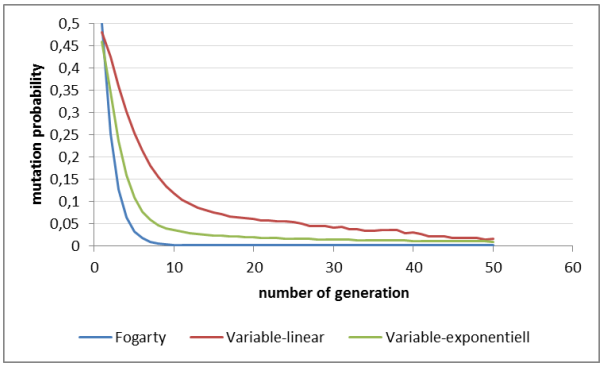


**Fig 12: Resulting mutation rate during optimization run at Schwefel's function.**

## 6. SUMMARY AND OUTLOOK

For a set of test scenarios and several test cases based on two test functions, it was shown for a multi-chromosome representation that decreasing mutation probability has a positive effect on GA performance, as observed by Hesser&Männer[6] or Fogarty [5]for single-chromosome representation.

We introduced chromosome fitness as the fitness value of a single chromosome in a multi-chromosome representation. Thereby we used each single chromosome for representing the different parts of the problem. Chromosome fitness determines the individual mutation rate for each chromosome: high mutation rate for chromosomes with worse fitness values and low mutation rate for chromosomes with good fitness values. Based on this, each chromosome can be optimized with individual increments towards its individual target.

Thereby the GA performance was faster or in a very good range in comparison to the scenarios with constant mutation rates. Compared to the Fogarty scenarios it also performed better.

For our approach it is also not necessary to hand-tune the control parameter anymore. Hinterding's approach reached this by using self-adapting strategy [8]. But in our approach there are no costs of self-adapting. Hinterding understood that as the used time to perform a good mutation rate based on the specific problem to be optimized. This is relevant for complex problems as well as for problems with a long startup phase (stabilizing) e.g. for optimization of production facilities.

In our future work we will test variable mutation rates based on chromosome fitness on a complex problem. This will be done on patients scheduling and sequencing problem in a university hospital with several hospital departments, each of them with individual targets and individual conditions. Based on several considered wards, the simulation model will have a long startup phase.

## 7. REFERENCES

[1] Holland, J. H. (1975). Adaption in natural and artificial systems. An introd. analysis with applications to biology, control, and artificial intelligence. Univ. of Michigan, Ann Arbor.

[2] DeJong, K. A. (1975). A nalysis of the behavior of a class of genetic adaptive. Ph.D. thesis, Univ. of Michigan.

[3] Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics Vol. 16, No. 1, 122–128.

[4] Schaffer, J. D., Caruana, R. A., Eshelman, L. J., & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the Third International Conference on Genetic Algorithms, 51–60.

[5] Fogarty, T. C. (1989). Varying the probability of mutation in the genetic algorithm. Proceedings of the 3rd International Conference on Genetic Algorithms, 104–109.

[6] Hesser, J.,&Männer, R. (1991). Towards an Optimal Mutation Probability for Genetic Algorithms. Proceedings of 1st workshop : Parallel problem solving from nature, 23–32.

[7] Bäck, T. (1992). Self-Adaption in Genetic Algorithms. Proceedings of the 1st European Conference on Artificial Life, 263–271.

[8] Hinterding, R. (1997). Self-adaptation using multi-chromosomes. Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, 87–91.

[9] Juliff, K. (1993). A multi-chromosome genetic algorithm for pallet loading. Proceedings of the 5th International Conference on Genetic Algorithms, 467–473.

[10] Cavill, R., Smith, S., & Tyrrell, A. (2005). Multi-Chromosomal Genetic Programming. Proceedings of Genetic and Evolutionary Computation Conference (GECCO), Washington D.C., 1753–1759.

[11] Davidor, Y. (1991). Genetic Algorithms And Robotics - A Heuristic Strategy For Optimization. World Scientific Publishing Co. Pte. Ltd., Singapur.

[12] Bäck, T. (1993). Optimal mutation rates in genetic search. Forrest, S. (Ed.), Proceedings of the 5th International Conference on Genetic Algorithms, 2–8.

[13] Ochoa, G., Harvey, I., & Buxton, H. (1999). Error thresholds and their relation to optimal mutation rates. Floreano, J. et al. (Eds.) Proceedings of the Fifth European Conference on Artificial Life Vol. 1674, 54–63.

[14] Pierrot, H. J.,&Hinterding, R. (1997). Using multi-chromosomes to solve a simple mixed integer problem. Australian Joint Conference on Artificial Intelligence, 137–146.

[15] Nissen, V.,&Biethahn, J. (1999). Ein Beispiel zu stochastischen Optimierung mittels Simulation und einem Genetischen Algorithmus. In Simulation als betriebliche Entscheidungshilfe. State ofthe Art und neuere Entwicklungen; mit 20 Tabellen, J. Biethahn, W.,Hummeltenberg, B., Schmidt, P.,Stähly T., & Witte, Eds. Physica-Verl, Heidelberg, 108–125.

[16] Schwefel, H. P. (1975). Evolutionsstrategien und numerische Optimierung. Dissertationsschrift, TechnischeUniversität Berlin.

[17] Mitchell, M. (1996). An introduction to genetic algorithms. MIT Press, Cambridge, London.

[18] Bäck, T. (1996). Evolutionary algorithms in theory and practice. Evolution strategies, evolutionary programming, genetic algorithms. Oxford Univ. Press, New York.

[19] Goldberg, D. E. (2004). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, Mass.

[20] Pohlheim, H. (2000). Evolutionäre Algorithmen. Verfahren, Operatoren und Hinweise für die Praxis. VDI-Buch. Springer, Berlin.

[21] Ochoa, G., Harvey, I., & Buxton, H. (2000). Optimal Mutation Rates and Selection Pressure in Genetic Algorithms. Proceedings of Genetic and Evolutionary Computation Conference, 315–322.