1 Statistical Inference as a Theoretical Foundation of Genetic Algorithms

Research site: GMD Learning Systems Laboratory
Research period: April 1,1993 - March 31, 1994

1.1 Goals

The objectives of this research are to develop a predictive theory of the Breeder Genetic Algorithm **BGA** and to solve Grand Challenge applications with the BGA. The aim of the theory is to identify measures which allow to control the BGA search most effectively. The research should lead to a scientific foundation of one of the key technologies in real world computing - problem solving by simulating evolution. Grand Challenge problems are important problems of science and society not yet solved. Some examples are cursiv handwriting, large transport optimization problems and protein folding.

1.2 Contents of the research

The current fashionable "theory" of genetic algorithm is based more on magic and belief than on science. Especially the "fundamental theorem of genetic algorithms" is either a tautology or wrong, depending on the interpretation. In contrast the theory of the BGA is based on classical population genetics and statistics. The key concepts are the *response to selection equation* and the *heritability. Regression techniques* and *decomposition of variance* are used to estimate the heritability. The power of the BGA will be demonstrated by solving difficult applications. The conducted research items have been as follows in fiscal year 1993

- Theory
- Optimization
- Discriminance analysis
- Synthesis of sigma-pi neural networks

In fiscal year 1993 most emphasis has been placed on the theory. Therefore the largest part of the report is devoted to the theory. Optimization and discriminance analysis are only shortly described. The synthesis of sigma-pi neural networks will be discussed in more detail.

1.3 Results

1.3.1 Theory of the breeder genetic algorithm

A. Introduction

Evolution of natural organisms is based on three major components - reproduction, variation and selection. Some reproductions of natural organisms occur with "failures" called mutations. A more systematic variation of the genetic material happens in sexual reproduction. Each parent contributes half of its genetic material to the offspring. This method of variation is called recombination. The offspring will be identical to the parents if the parents are genetically equal.

Variation is necessary to allow selection to work. Selection in nature is very difficult to define precisely. The term was introduced by Darwin [7] very informally. "*The preservation of favourable variations and the rejection of injurious variations, I call Natural Selection.*" But how can an observer predict which are the favorable variations? The favorable variations are the variations which are preserved! The variations can only be judged after they have competed in the "struggle for life." Natural selection is no independent force of nature, it is the result of the competition of natural organisms for resources.

In contrast, in the science of breeding the above problem does not exist. The selection is done by human breeders. Their strategies are based on the assumption that mating two individuals with high fitness more likely produces an offspring of high fitness than two randomly mating individuals. The *Breeder Genetic Algorithm* **BGA** introduced in [23] is based on the science of breeding. The science is part of applied statistics. A major component is the regression of parent and offspring.

In this report we first introduce the basic concepts, the *response to selection* and the *heritability*. Then the major theoretical results will be summarized. They have been obtained for a rather simplified additive genetic model. Nevertheless its behavior is surprisingly complex. The model has the same importance as the "ideal gas" in thermodynamics. The model needs already five parameters. In the last sections the statistical measures will be applied to complex fitness functions.

B. Natural vs. artificial selection

The theoretical analysis of evolution centered in the last 60 years on understanding evolution in a natural environment. It tried to model *natural selection*. But modelling natural selection is difficult. Natural selection is not an independent force of nature, it is the result of the competition of the organisms in their struggle for life. Usually biologists introduce another term, the fitness of an individual which is defined as the number of offspring of that individual. This fitness definition cannot be used for prediction. It can only be measured after the individual is not able to reproduce any more.

Artificial selection as used by breeders is seldom investigated in textbooks on evolution. It is described in more practical books aimed for the breeders. We believe that this is a mistake. Artificial selection can be seen as a controlled evolution experiment. It can be used to isolate and understand specific aspects of evolution. Individuals are selected by the breeder according to some trait. The prediction of the outcome of a breeding experiment plays a major role for the breeder.

Darwin recognized the importance of artificial selection. He devoted the whole first chapter of his book to artificial selection by breeders. In fact, artificial selection independently done by a number of breeders served as a model for natural selection. Darwin wrote: "I have called this principle by the term Natural Selection in order to mark its relation to man's power of selection."

In this section we will analyze artificial selection by methods found in [10], [5] and [3]. A mathematically oriented book on quantitative genetics and natural selection is [6]. Proportionate selection which is used by the simple genetic algorithm [15] can be analyzed by the same methods. A detailed comparison can be found in [23] [22].

The change produced by selection that mainly interests the breeder is the *response to selection*, which is symbolized by R. R is defined as the difference between the population mean fitness of generation $t+1$ and the population mean of generation t. $R(t)$ measures the expected progress of the population.

$$R(t) = M(t+1) - M(t) \tag{1}$$

Breeders measure the selection with the *selection differential*, which is symbolized by S. Let the selected parents be denoted by P_s. Then $S(t)$ is given by

$$S(t) = M(t, P_s) - M(t) \tag{2}$$

$S(t)$ is defined as the difference between the mean fitness of the selected parents and the mean fitness of the population. The breeder tries to predict $R(t+1)$ from $S(t)$. This regression is shown in figure 1. The curves represent the fitness distribution of the phenotypes at generations t and $t+1$.

Breeders often use *truncation selection* or *mass selection* as shown in figure 1. In truncation selection with threshold T, the T % best individuals will be selected as parents. T is normally chosen in the range 50% to 10%.

The prediction of the response to selection starts with

$$R(t) = b_t \cdot S(t) \tag{3}$$

The breeder either measures b_t in previous generations or estimates b_t by different methods. A method based on the regression of parents to offspring will be explained later. b_t is called *realized heritability* in quantitative genetics. It is normally assumed that b_t is constant for a certain number of generations. This leads to

$$R(t) = b \cdot S(t) \tag{4}$$

There is no genetics involved in this equation. It is simply an extrapolation from direct observation. The response to selection is the product of the heritability and

T	80 %	50 %	40 %	20 %	10 %	1 %
I	0.34	0.8	0.97	1.2	1.76	2.66

Table 1: Selection intensity.

the selection differential. The prediction of just one generation is only half the story. The breeder (and the GA user) would like to predict the cumulative response R_s for s generations of his breeding scheme.

$$R_s = \sum_{t=1}^{s} R(t) = \sum_{t=1}^{s} b \cdot S(t) \tag{5}$$

In order to compute R_n a second equation is needed. In quantitative genetics, several approximate equations for $S(t)$ are proposed [3], [10]. Unfortunately these equations are only valid for *diploid* organisms. Diploid organisms have two sets of chromosomes. Most genetic algorithms use one set of chromosomes, i.e. deal with *haploid* organisms. Therefore, we can only apply the research methods of quantitative genetics, not the results.

If the fitness values are normal distributed, the selection differential $S(t)$ in truncation selection is approximately given by

$$S = I \cdot \sigma_p \tag{6}$$

where σ_p is the phenotypical standard deviation. I is called the *selection intensity.* The formula is a feature of the normal distribution. A derivation can be found in [3]. In table 1 the relation between the truncation threshold T and the selection intensity I is shown. A decrease from 50 % to 1 % leads to an increase of the selection intensity from 0.8 to 2.66 only.

If we insert (6) into (4) we obtain the well-known *response to selection equation* [10].

$$R(t) = b \cdot I \cdot \sigma_p(t) \tag{7}$$

The science of artificial selection consists of estimating b and $\sigma_p(t)$. The estimates depend on the fitness function. Applications can be found in [23], [22]. Because of shortage of space we will just summarize the major theoretical results.

C. Summary of the major theoretical results

In this section we will just survey the results which can be found in [20],[22],[24],[2]. They are valid for fitness functions with additive gene effects. Let n denote the number of genes, N the size of the population.

We first consider populations with recombination and no mutation. Any finite population of size N will converge to a single genotype, even if selection is not applied. This effect is called *genetic drift.* The number of generations until convergence GEN_e is surprisingly low.

$$GEN_e \propto N \cdot ln(n) \qquad no\_sel, rec, no\_mut \tag{8}$$

We now turn to truncation selection. If the size N of the population is larger than the *critical popsize* N^*, the minimum popsize to converge to the optimum with high probability, then we have

$$GEN_e \propto \frac{\sqrt{n}}{I} \qquad trunc\_sel, rec, no\_mut, N \geq N^* \tag{9}$$

Note that GEN_e is independent of N. The estimation of the critical popsize is very difficult. The dependence of N^* from I is nonlinear. Simulations have shown that N^* increases for large selection intensities and for small selection intensities [24]. For small selection intensities this behavior seems surprisingly. But the reason is the *genetic drift* which reduces the variance of the population. We conjectured

$$N^* = \sqrt{n} \cdot ln(n) \cdot f_1(p_0) \cdot f_2(I) \tag{10}$$

Proportionate selection as used in the simple GA [15] selects too weak when the variance of the population gets small. The expected number of generations $GEN_{1-1/n}$ until the favorable allele is distributed in the population with probability of $1 - 1/n$ is given by

$$GEN_{1-1/n} \propto n \cdot ln(n) \qquad prop\_sel, rec, no\_mut, N \gg 0 \tag{11}$$

This number is much larger than with truncation selection.
The analysis of recombination in small populations is difficult. We have shown in [24] the results in phase diagrams relating the posize and GEN_e. The phase diagrams can be divided into two areas. The border is given by the critical popsize N^*.
We now turn to populations using only mutation. Mutation is a random search operator especially efficient in small populations. The most important result concerns the mutation rate. The mutation rate is defined as the probability of mutating a gene.

Rule of thump: *The mutation rate $m = 1/n$ where n is the size of the chromosome is almost optimal [21].*

For the above mutation rate the expected number of generations GEN_{opt} until the optimum is found has been computed for the $(1+1)-$strategy (one parent, one offspring; the better of the two survives).

$$GEN_{opt} \propto n \cdot ln(n) \qquad sel, no\_rec, mut, N = 2 \tag{12}$$

Mutation in large population is inefficient. The scaling remains the same as for $N = 2$. But it is still twice as efficient as proportionate selection with recombination [24].

$$GEN_{1-1/n} \propto n \cdot ln(n) \qquad sel, no\_rec, mut, N \gg 0 \tag{13}$$

For binary fitness functions, populations using either recombination or mutation are able to locate the optimum. Moreover, the asymptotic order of the number of trials needed (FE_{opt}), seems to be the same, namely $O(n \cdot ln(n))$. For recombination this number is obtained by multiplying GEN by the critical popsize N^*. Therefore the question which of the two operators is more efficient is difficult to answer. The comparison needs an exact expression for N^*, which we have not yet obtained. But we can easily make a qualitative comparison. The major difference between mutation and recombination is their dependence on p_0, the percentage of the desired allele in the initial population.

Let us take $p_0 = 1 - 1/n$ as example. Then just one bit of a chromosome is wrong on the average. Mutation will need about $O(n)$ trials to change the incorrect bit. Uniform crossover of two strings, each with one bit wrong, will generate the optimum string with probability 1/4, independent of the size of the problem. Therefore the critical popsize N^* is also independent of n. Thus recombination is much more efficient than mutation. But the determination of the exact N^* is also difficult in this simple case. It will need on the average 4 trials to generate the optimum. But the probability that a popsize of 4 will not generate the optimum is $0.75^4 = 0.31$. It needs 16 trials in order to obtain the optimum with 99% probability.

If we take $p_0 = 1/n$ the situation is reversed. Here only one bit is correct on the average. Now mutation is much more efficient than recombination which needs a huge popsize in order to locate the optimum. It is obvious that mutation is more successful than recombination when far from the optimum. Recombination has too few building blocks to generate better offspring. But recombination is more effective than mutation near the optimum. Here the success of a mutation is the lowest.

A more detailed comparison between mutation and recombination, also by means of a competition between populations can be found in [22]. We now turn to general fitness functions.

D. Heritability and regression

The first theorem connects the realized heritability $b_t = R(t)/S(t)$ with the regression coefficient between *midparent* and offspring. Let $x_i(t), x_j(t)$ be the phenotypic values of parents i and j, then

$$x_m(t) = \frac{x_i(t) + x_j(t)}{2}$$

is called the midparent value. Let the stochastic variable X_m denote the midparent value.

Theorem 1 *Let $X(t) = (x_1(t), \ldots x_N(t))$ be the population at generation t, where x_i denotes the phenotypic value of individual i. Assume that an offspring generation*

$X'(t+1)$ is created by random mating, without selection. If the regression equation

$$x'_{ij}(t+1) = a(t) + b_{X'X_m}(t) * \frac{x_i(t) + x_j(t)}{2} + \epsilon_{ij} \tag{14}$$

with

$$E(\epsilon_{ij}) = 0$$

is valid, where x'_{ij} is the offspring of x_i and x_j, then

$$b_{X'X_m}(t) \approx b_t \tag{15}$$

The proof can be found in [24]. The theorem says that the realized heritability $R(t)/S(t)$ is equal to some regression coefficient of midparent and offspring.

The importance of regression for estimating the heritability was discovered by Galton and Pearson. They used scatter diagrams of midparent and offspring. The computation of the regression coefficient was done rather intuitively [13]. The problem of computing a good regression coefficient is solved by the theorem of Gauss-Markov. We just cite the theorem. The proof can be found in any textbook on statistics [25].

Theorem 2 *A good estimate for the regression coefficient of midparent and offspring is given by*

$$b_{X'X_m}(t) = \frac{cov(x'(t), x_m(t))}{var(x_m(t))} \tag{16}$$

The covariance of X' and X_m is defined by

$$cov(x'(t), x_m(t)) = \frac{1}{N} \sum_i (x'_i(t) - av(x'(t))) * (x_{m_i}(t) - av(x_m(t)))$$

where av denotes the average and var the variance. Closely related to the regression coefficient is the correlation coefficient $r_{X'X_m}$. It is given by

$$r_{X'X_m}(t) = b_{X'X_m}(t) * (\frac{var(x_m(t))}{var(x'(t))})^{1/2}$$

The above theorem enables us to estimate the heritability by a second method. It works as follows. For a large sample population X the offspring have to be created by random mating. Then the regression coefficient $b_{X'X}$ can be computed by equation 16. This procedure is more robust than dividing $R(t)$ by $S(t)$. First, it works also in the case of small selection intensity, and second, the trustworthiness of the computation can be estimated by statistical techniques.

By the above method an average value for the heritability is computed. The average is taken over the whole domain. For the breeder genetic algorithm we decided to proceed slightly differently. The regression coefficient is only computed for the *selected parents* and their offspring. This local approximation makes it possible to compute regression coefficients which depend on the given population and the local fitness landscape.

The next theorem shows the connection between *midparent* and *parent* regression.

Theorem 3 *Midparent and parent regression are connected by*

$$b_{X'X}(t) = 0.5 \cdot b_{X'X_m}(t)$$

$$r_{X'X}(t) = \sqrt{\frac{1}{2}} r_{X'X_m}(t)$$

For purely additive gene effects we have $b_{X'X_m} = 1$ *and* $r_{X'X_m}(t) = \sqrt{\frac{1}{2}}$

Proof 1 *We have*

$$cov(x'(t), x_m(t)) = cov(x'(t), x(t))$$

$$var(x_m(t)) = var(0.5(x_1(t) + x_2(t)) = 0.5 \cdot var(x(t))$$

From (16) the first part of the theorem is obtained. A rigorous proof of the second part will be given in a forthcoming paper.

The above theorem shows that the maximum correlation coefficient for midparent and offspring is $\sqrt{0.5}$. For parent and offspring the maximum is 0.5. This result one would intuitively expect. About half of the genes of each parent are transferred to the offspring. The theorem assumes that the parents are different. If the parents are phenotypically and genetically equal, then the correlation coefficient will be 1.

In the next section we will apply the above statistical measures for different fitness functions to analyse the BGA.

E. Application of the breeding theory

In this section we will show how to apply the theory. We will use continuous and binary fitness functions. We start with continuous functions. The BGA for continuous functions has been described in [23]. It uses a floating point representation. In our simulations we will use the genetic operator *discrete recombination.* It chooses with probability 0.5 for each variable one of the two floating point values of the parents.

As a first example we take the minimization of the hypersphere. In figure 2 the two different estimates for the heritability are shown for each generation. For the simulation a truncation threshold of 0.5 was applied. Both heritability estimates oscillate around 1 as predicted. The correlation coefficient stays at about 0.5. This is less than the maximum value of $\sqrt{0.5}$. The reason for this difference is the selection. The selection reduces the variance of the parents and therefore the correlation coefficient.

We just report the results for a simulation run without selection. In this case the $R(t)/S(t)$ estimator cannot be used because S(t) is about 0. The regression coefficient can be computed as usual and remains 1. Furthermore the correlation coefficient is about $\sqrt{0.5}$ as predicted by the theory.

The above results are not restricted to simple unimodal functions. As next example we take the highly multimodal function which is known as Schwefel's function $F7$ [23].

$$F_7 = \sum_1^n -x_i \, sin\left(\sqrt{|x_i|}\right) \qquad -500 \leq x_i \leq 500 \tag{17}$$

The theory predicts that the multimodality of this function can be considered more or less as noise for the BGA. It should have no major influence on the regression coefficient. Indeed, with random mating, the regression coefficient is 1 and the correlation coefficient between midparent and parent is about $\sqrt{0.5}$, just as for the hypersphere. Figure 3 shows a real BGA simulation run with selection, recombination *and* mutation. One clearly observes that the search is first driven by recombination, then by mutation. From generation 20 on, the regression coefficient substantially differs from the ratio estimator $R(t)/S(t)$. Now the search is mainly driven by the random operator mutation. The BGA mutation scheme is described in [23].

Next we turn to binary functions. We take as examples

- ONEMAX(n)
- PLATEAU(20,3)
- DECEP(10,3)

$ONEMAX$ gives the number of 1's in the string, $PLATEAU(20,3)$ has a string length of 60. An increase in fitness is allocated if three consecutive bits at loci 1,3,6,.. are 1's. In each case, the fitness is increased by 3. $DECEP(10,3)$ is the deceptive function defined by Goldberg [16].

In figure 4 the results of a BGA run are shown for $ONEMAX(64)$ with a truncation threshold of $T = 0.5$, uniform crossover but without mutation. The two heritability estimates coincide fairly well. They are about 1, as predicted. The correlation coefficient is about 0.5 till generation 14. This is less than the correlation coefficient without selection, which is $\sqrt{0.5}$. At the end of the run the correlation coefficient increases. This behavior indicates that the genotypes of the selected parents become too similar. Therefore the offspring is equal to the parents.

Our next example is the PLATEAU function. We will discuss PLATEAU(20,3) and PLATEAU(20,5). PLATEAU(20,5) has a plateau of size 5, therefore it is more difficult to optimize. Without selection the regression coefficients for the two functions are about 0.7 and 0.4, the correlation coefficients are about 0.5 and 0.3. In figure 5 we have used a truncation threshold of $T = 0.5$. For both functions the regression coefficients are substantially higher than without selection. This indicates that selection is very effective for this fitness function. But note that the realized heritability $R(t)/S(t)$ is considerably smaller.

The last example is the deceptive function DECEP(10,3). This function is called deceptive, because the search is guided into the local optimum $(0,0,0)$. The global optimum is at $(1,1,1)$. Without selection, the regression coefficient is about 0.5 and the correlation coefficient about 0.35. This is shown in figure 6.

The behavior radically changes with selection. If selection is applied, both the regression coefficient and the ratio estimator become erratic and half of the time negative. This shows that the fitness function deceives the crossover operator. The operator is creating offspring which are not correlated to the parents.

To summarize this section: *The theory presented in the previous section is especially applicable for continuous functions. For many continuous fitness functions the regression coefficient will be* 1, *the maximum possible. For binary functions the regression coefficient and the realized heritability give useful information about the complexity of the fitness landscape and how to guide the breeding programme.*

F. A short history of population genetics

The theory presented in the previous sections is an adaptation of classical concepts of population genetics and statistics to genetic algorithms. Therefore a review about the most important developments seems appropriate.

Genetics represents one of the most satisfying applications of statistical methods. Galton and Pearson found at the end of the last century a striking empirical regularity. On the average a son is halfway between his father and the overall average height for sons. They used data from about 1000 families. In order to see this regularity Galton and Pearson invented the scatter diagram, regression and correlation [13].

Independently Mendel found some other striking empirical regularities like the reappearance of a recessive trait in one-fourth of the second generation hybrids. He made up a chance model involving what are now called *genes* to explain his rules. He conjectured these genes by pure reasoning - he never saw any.

At first sight, the Galton-Pearson results look very different from Mendel's, and it is hard to see how they can be explained by the same biological mechanism. Indeed Pearson wrote an article in 1904 claiming that his results cannot be explained by Mendel's laws. About 1920 Fisher, Wright and Haldane more or less simultaneously recognized the need to recast the Darwinian theory as described by Galton and Pearson in Mendelian terms. They succeeded in this task, but unfortunately much of the original work is very difficult to follow. The difficulty lies in the concept of *additive genetic variance* and its connection to *heritability*. We will explain the additive genetic variance and its application in the Breeder Genetic Algorithm in a forthcoming paper. Here we just cite from [9]. "The additive genetic variance in any character measures, approximately, that part of the total variance that can be accounted for by genes responsible for that character, freed from the further variatation caused by interaction between these genes. Since it is a gene that a parent passes on to his offspring, rather than a genotype, at any locus, it is only the additive part of the genetic variance that has evolutionary significance."

We are currently trying to adapt the concept of additive genetic to our breeder genetic algorithm.

In summary: *There are three different methods to estimate the heritability. The first one is $b_t = R(t)/S(t)$. This number is called the realized heritability. The second one is the regression coefficient offspring to midparent of Galton and Pearson. The third estimate is the quotient of the additive genetic variance divided by the total variance V_A/V_p.*

1.3.2 Optimization

The theoretical results have been used to design and control the breeder genetic algorithm for continuous function optimization. We have shown the following theorem [23].

Theorem 4 *Let a unimodal fitness function F of n variables be given. Let the initial solution be $||d_0||$ away from the optimum. Then in order to locate the optimum with a precision of ϵ the BGA using a popsize of two and mutation only will need about*

$$FE = c \cdot n \cdot ln \frac{||d_0||}{\epsilon} \tag{18}$$

function evaluations.

This result shows that the BGA scales linearly in the number of variables. We conjecture that the above scaling is optimal for search procedures which do not use derivatives of the function.

Our theoretical results indicate that the above theorem should also be valid for many multimodal functions. This has been confirmed by simulations. At this moment the BGA scales better than any other known genetic algorithm.

Knowing that the scaling is optimal we will reduce the scaling factor c in the future. This will be done by introducing a *multiresolution search.* This will be achieved by a competition between populations. Different populations search the space by different strategies e.g. large steps or small steps. The populations which are more successful than the average will increase.

1.3.3 Discriminance analysis

The design for using the BGA for discriminance analysis has been completed. Preliminary investigations have been done using different fitness functions. The results will be reported later.

1.3.4 Sigma-pi neural networks

A. Introduction

Genetic programming has been successfully used to evolve computer programs for solving many interesting problems in artificial intelligence and artificial life [4, 18, 19]. Similar to usual genetic algorithms (GAs), genetic programming (GP) starts with a population of randomly generated individuals. Each individual is a program that, when executed, is the candidate solution to the problem. These programs are expressed as parse trees, or LISP S-expressions. Fitness proportionate selection and crossover are used to produce increasingly fitter populations of computer programs.

While most GAs use binary strings of fixed size, GPs use structured representations of variable length. This is important because it is particularly suited to problems in which the optimal underlying structure must be discovered. One problem with the variable length is that the program size may grow without bound. For example, Kinnear [18] reports that all but a very few of evolved solutions to his sorting problems were so large as to defy any human understanding of them. Tackett [27], in his application of genetic programming to image processing tasks, oberserves that the size and complexity of trees grows without performance improvement.

In [29, 28] we introduced the breeder genetic programming (BGP) that employs Occam's razor in its fitness measure to evolve optimal or minimal size multilayer perceptrons. We apply here the BGP method to synthesize sigma-pi neural networks. Unlike multilayer perceptrons, sigma-pi networks use product units as well as summation units to build higher-order terms. In section B we illustrate the usefulness of higher-order terms and show how the sigma-pi networks can be used to represent higher-order networks. Results of simulation will be reported later.

B. Sigma-Pi Neural Networks

To motivate the approach, we start with a brief description of multilayer neural networks. Multilayer perceptrons are feedforward networks with one or more layers of nodes between the input and output units. These additional layers contain hidden units that are not directly connected to both the input and output units. The input-output relation of the units is given in these networks by weighted sum of inputs

$$u_i = \sum_{j \in R(i)} w_{ij} x_j \tag{19}$$

where w_{ij} is the connection weight from unit j to unit i and $R(i)$ denotes the receptive field of unit i. The total input is then transferred to upper layer units by a nonlinear activation function f, e.g. a threshold function:

$$y_i = f(u_i) = \begin{cases} +1 & \text{if } u_i > \theta_i \\ -1 & \text{otherwise} \end{cases} \tag{20}$$

where θ_i is a threshold.

A commonly adopted architecture consists of one hidden layer with full connectivity between neighboring layers. This structure has been very successful for many applications. However, they have some weaknesses:

1. The full connectivity between layers and often only between neighboring layers try to find a prediction over the full input space. This is not necessarily a good strategy. A particular task might not contain a good predictor for the full input space, but might contain functions capable of good prediction on specified regions of input space.

2. They are especially appropriate to approximating additive functions, since they employ linear combinations of inputs. However, the multilayer perceptrons cannot approximate efficiently if there are high order interactions between the inputs. Adding additional hidden layers may help to extend the representational capacity of the network.

In [29, 28]. we use genetic programming to construct a problem-specific architecture whose size and depth are adapted flexibly during evolution. The method allows partial connectivity and direct connections between non-neighboring layers to find a parsimonious architecture. In this architecture input units can be connected directly to output units. Although the method turned out to be useful to find parsimonious networks, the simulation results and the analysis of the landscapes suggested that the representation scheme does not scale well on parity-like problems.

To improve the scaling property, we extended the function set of the genetic programming to include pi-units as well as the sigma-units. While a sigma-unit calculates a sum of weighted inputs, a pi-unit builds a *product* of weighted inputs:

$$u_i = \prod_{j \in R(i)} v_{ij} x_j. \tag{21}$$

Here v_{ij} is the connection weight from unit j to unit i and $R(i)$ denotes the receptive field of unit i. The resulting total input is propagated to upper layer units by an activation function chosen depending on applications.

The pi-units has been suggested earlier in the neural network community [8, 11, 26] and employed in polynominal networks [12, 17] and higher-order networks [1, 14].

A higher-order neuron of order k has an input-output relation given by

$$\begin{aligned} y &= f(u), \\ u &= w_0 + \sum_i w_i^{(1)} x_i + \sum_{i_1} \sum_{i_2} w_{i_1 i_2}^{(2)} x_{i_1} x_{i_2} + \cdots \\ &\quad + \sum_{i_1} \sum_{i_2} \cdots \sum_{i_k} w_{i_1 \cdots i_k}^{(k)} x_{i_1} \cdots x_{i_k} \end{aligned} \tag{22}$$

where all indices $i_1, ..., i_m$ in $w_{i_1 \cdots i_m}^{(m)}$ are assumed to take different values satisfying $i_1 < i_2 < \cdots < i_m$.

Since the k-th order term consists of a linear weighted sum over k-th order products of inputs, we can rewrite it using pi-units:

$$\begin{aligned}
T^{(k)} &= \sum_{i_1}\sum_{i_2}\cdots\sum_{i_k} w^{(k)}_{i_1\cdots i_k} x_{i_1}\cdots x_{i_k} \\
&= \sum_{i_1}\sum_{i_2}\cdots\sum_{i_k} w^{(k)}_{i_1,\ldots,i_k} g\left(\prod_{i=i_1}^{i_k} x_i\right) \\
&= \sum_{i_1}\sum_{i_2}\cdots\sum_{i_k} w^{(k)}_{i_1,\ldots,i_k} g\left(\prod_{i=i_1}^{i_k} v_i x_i\right) \\
&= \sum_{(i_1,i_2,\ldots,i_k)} w^{(k)}_{i_1,\ldots,i_k} P^{(k)}, \qquad (23)
\end{aligned}$$

where

$$P^{(k)} = P^{(k)}_{i_1,i_2,\ldots,i_k} = g\left(\prod_{i=i_1}^{i_k} v_i x_i\right) \qquad (24)$$

assuming

$$g(u) = u \text{ and } v_i = 1. \qquad (25)$$

The higher-order terms can be again used as building blocks which are able to capture a high-order correlational structure of the data. In particular, by building a sigma unit which has as input various higher-order terms, we can construct a higher-order network of sigma-pi units:

$$y_i = f_i(u_i) = f_i\left(\sum_k w_k T^{(k)}\right) \qquad (26)$$

The problem in using higher-order networks is that the number of terms explodes with the problem size; the number of parameters necessary for specifying an order k neuron is

$$r_k = \sum_{i=0}^{k} {}_nC_i, \qquad (27)$$

because $w^{(m)}_{i_1\cdots i_m}$ have ${}_nC_m$ components. Here n is the total number of inputs and ${}_nC_m$ are the binomial coefficients. As an example, an order 4 neuron has $2^4 = 16$ parameters as shown in Figure 7.

To avoid the combinatorial explosion, a method is needed to discover and combine useful terms and to eliminate non-essential terms. This will be done with the breeder genetic algorithm. The sigma-pi networks are represented as a set of m trees, where m is the number of output units. Each tree has an arbitrary number of subtrees.

The major problem of genetic programming is the operator *exchange of subtrees*. This operator is not a crossover operator in the usual sense. It violates the requirement that a crossover of two genetical equal parents creates a genetical identical offspring. Therefore the theory of the BGA has to be extended for this operator.

References

[1] Shun-Ichi Amari. Dualistic geometry of the manifold of higher-order neurons. *Neural Networks*, 4:443–451, 1991.

[2] G. Asoh and H. Mühlenbein. On the mean convergence time of genetic populations without selection. Technical report, GMD, Sankt Augustin, 1994.

[3] M. G. Bulmer. *"The Mathematical Theory of Quantitative Genetics"*. Clarendon Press, Oxford, 1980.

[4] Robert James Collins. *Studies in Artificial Evolution.* PhD thesis, University of California, Los Angeles, 1992.

[5] J. F. Crow. *Basic Concepts in Population, Quantitative and Evolutionary Genetics.* Freeman, New York, 1986.

[6] J. F. Crow and M. Kimura. *An Introduction to Population Genetics Theory.* Harper and Row, New York, 1970.

[7] Ch. Darwin. *The Origins of Species by Means of Natural Selection.* Penguin Classics, London, 1859.

[8] Richard Durbin and David E. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1:133–142, 1989.

[9] W.J. Ewens. On the role of mathematical statistics in population genetics. In K.V. Mardia, editor, *The Art of Statistical Science*, pages 233–260. Wiley, 1992.

[10] D. S. Falconer. *Introduction to Quantitative Genetics.* Longman, London, 1981.

[11] J. A. Feldman and D. H. Ballard. Connectionistmodels and their properties. *Cognitive Science*, 6:205–254, 1982.

[12] Jürgen Franke. On the functional classifier. In *Proceedings of the First International Conference on Document Analysis and Recognition*, 1993.

[13] D. Freedman, R.Pisani, R. Purves, and A. Adhikkari. *Statistics second edition.* W.W. Norton, New York, 1991.

[14] C. Lee Giles and Tom Maxwell. Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, 26(23):4972–4978, 1987.

[15] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, 1989.

[16] D.E. Goldberg. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[17] A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(4):364–378, 1971.

[18] K. E. Kinnear Jr. Generality and difficulty in genetic programming: Evolving a sort. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann.

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

[20] H. Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, San Mateo, 1991. Morgan-Kaufman.

[21] H. Mühlenbein. How Genetic Algorithms Really Work: Mutation and Hill-climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature*, pages 15–26, Amsterdam, 1992. North-Holland.

[22] H. Mühlenbein and D. Schlierkamp-Voosen. Analysis of Selection, Mutation and Recombination in Genetic Algorithms. *Neural Network World*, 3:907–933, 1993.

[23] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: Continuous Parameter Optimization. *Evolutionary Computation*, 1:25–49, 1993.

[24] H. Mühlenbein and D. Schlierkamp-Voosen. The science of breeding and its *application to the breeder genetic algorithm.* Technical report, GMD, Sankt Augustin, 1994.

[25] C.R. Rao. *Linear Statisticcal Inference and Its Application.* Wiley, New York, 1973.

[26] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. A general framework for parallel distributed processing. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume I, pages 45–76. MIT Press, 1986.

[27] W. A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 303–309, San Mateo, CA, 1993. Morgan Kaufmann.

[28] Byoung-Tak Zhang and Heinz Mühlenbein. Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 1993. (forthcoming).

[29] Byoung-Tak Zhang and Heinz Mühlenbein. Genetic programming of minimal neural nets using Occam's razor. In Stephanie Forrest, editor, [illegible]*ings of the Fifth International Conference on Genetic Algorithms*, pages 342–349. Morgan Kaufmann, San Mateo, 1993.

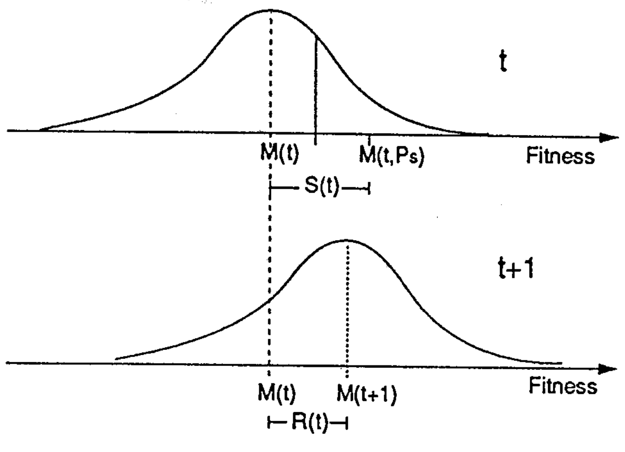

Figure 1: Response to truncation selection

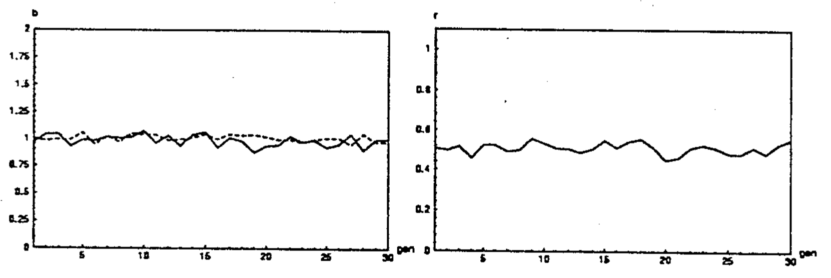

Figure 2: Heritability estimates and correlation coefficient (N=1024,T=0.5)

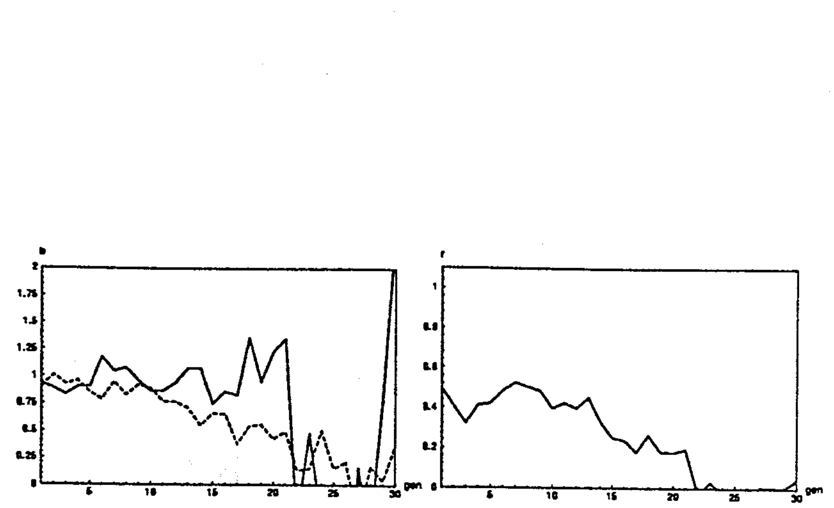
Figure 3: Heritability estimates with mutation and recombination ($N = 256$). The correlation coefficient r drops to zero. The regression coefficient (solid line) and the ratio estimator are almost equal at the beginning. Then the ratio $R(t)/S(t)$ goes to zero whereas the regression coefficient remains high till generation 22.

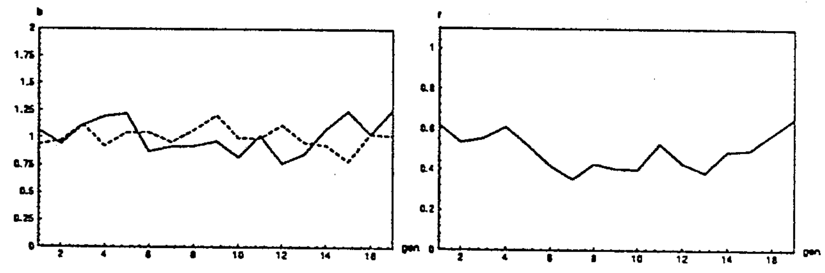
Figure 4: Heritability and correlation estimates with recombination for ONEMAX(64) ($N = 128, T = 0.5$)

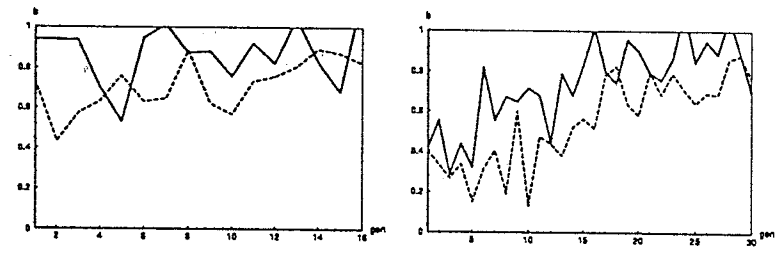
Figure 5: Regression coefficient - and $R(t)/S(t)$ - - for PLATEAU(20,3) and (20,5)

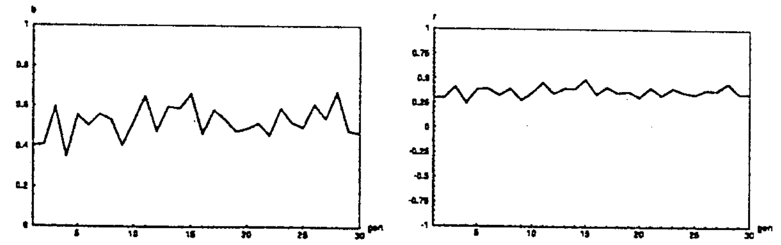
Figure 6: Heritability and correlation estimates with recombination for DECEP(10,3) ($N = 256, T = 0.5$)

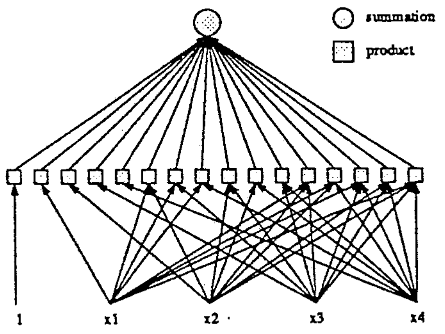

Figure 7: *An order 4 neuron*