Genetic Self–Learning

Frank Hoffmeister* Thomas Bäck†

University of Dortmund · Department of Computer Science XI
P.O. Box 50 05 00 · D–4600 Dortmund 50 · Germany

Abstract

Evolutionary Algorithms are direct random search algorithms which imitate the principles of natural evolution as a method to solve adaptation (learning) tasks in general. As such they have several features in common which can be observed on the genetic and phenotypic level of living species. In this paper the algorithms' capability of adaptation or *learning* in a wider sense is demonstrated, and it is focused on *Genetic Algorithms* to illustrate the learning process on the population level (first level learning), and on *Evolution Strategies* to demonstrate the learning process on the meta–level of strategy parameters (second level learning).

Introduction

There is no doubt that the search process of natural evolution is a powerful mechanism for improving living beings on our planet by performing a random search in the space of possible DNA-sequences. Without the existence of a system capable of evolution — i.e. possessing the properties of a metabolism, self–reproduction, multistability, selection, and mutability — no life of the quality surrounding us could have emerged within the "short" period of 4 billion years as noted by Eigen (1971) and Ebeling and Feistel (1982).

Due to this newer knowledge about the qualities of natural evolution, some researchers tried to use the basic mechanisms of evolution as a basis of optimum–seeking techniques in case of vast search spaces. Without mentioning all important approaches, only the very early work of Bremermann (1965, 1968) and Fogel, Owens, and Walsh (1966) is stressed here besides the nowadays well-known and in most cases broadly accepted techniques. Most notably among them are the *Genetic Algorithm* (GA) and *Classifier System* as developed by Holland (1975, 1986a, 1986b, 1987) and Booker et al. (1989), the *Evolution Strategy* (ES) by Rechenberg (1973) and Schwefel (1975, 1977, 1981, 1987, 1988, 1989), and the *Genetic Programming Paradigm* by Cramer (1985) and Koza (1987, 1989, 1990).

In any of these approaches a population of individuals is maintained. This population is able to adapt to a given (static or dynamically changing) environment by randomized processes of reproduction, sexual recombination, and mutation. Through an environmental feedback individuals get a quality information (fitness), and the selection process favours the individuals of higher quality to survive (even the reproduction processes within these algorithms often favour higher quality structures). Thus, during the evolution process the average quality of the population increases, hopefully leading to an optimum point.

Algorithms following this general approach have been summarized under the term *Evolutionary Algorithms* by Hoffmeister and Bäck (1991, 1991b). It is important here to note that the approaches are mainly differing in the structure of the individuals the algorithm is working upon. The recombination and mutation operators are generally influenced by the basic structure, and the unifying and generally best selection and reproduction strategy has not been developed yet as can be seen from different approaches by Holland (1975), Baker (1985, 1987), Whitley (1989), and Hoffmeister and Bäck (1990).

With respect to the complexity of the structures the algorithms are working on, a hierarchy as presented in table 1 can be constructed, where P_{LISP} denotes the set of syntactically correct LISP–programs, $\mathcal{P}(S)$ is the power set of a set S, l denotes the individ-

* `iwan@ls11.informatik.uni-dortmund.de`
† `baeck@ls11.informatik.uni-dortmund.de`

Structures		Terminology
Computer programs	P_{LISP}	Genetic Programming Paradigm
Rules of a rule–based system	$\mathcal{P}(\{0,1,\#,\vert,\ldots\}^l)$ $\{0,1,\#,\vert,\ldots\}^l$	Classifier–Systems (Pittsburgh-approach)[1] Classifier–Systems (Michigan-approach)[2]
Real–valued vectors	$\mathbb{R}^{2n+n/2\cdot(n-1)}$	Evolution Strategy
Bitstrings	$\{0,1\}^l$	Genetic Algorithm

Table 1: Hierarchy of structures worked upon by Evolutionary Algorithms

uals' length, and n is the dimension of the objective function and the corresponding number of mutation-rates for each object variable in an ES. From the top to the bottom of table 1 the amount of existing work on the actual topic increases as the theoretical and practical treatability of the structures increases due to their smaller degree of complexity.

Evolutionary Algorithms which model mechanisms from natural evolution are capable of adaptation to an unknown environment by maximizing the average fitness of the modelled population over time. In general, this corresponds to finding or *learning* of an optimal setting of some control parameters or of an internal representation of some reality in order to accomplish a given task. As such, Evolutionary Algorithms can be seen as an alternative approach in the context of machine learning.

According to a classification of machine learning strategies presented by Michalski (1986) Evolutionary Algorithms realize *learning by induction*, i.e. they work by some generalization of the input information. More exactly and in contrast to concept acquisition (learning by examples) this works by observation and discovery and provides the most advanced and complicated learning technique when it is compared to *rote learning*, *learning by instruction*, *learning by deduction*, and *learning by analogy*. Even humans are assumed to learn by inductive guessing general regularities from observed examples. Of course, these guesses are often too general and have to be corrected later on as described by Holland et al. (1987).

A further property of these algorithms which provides a high algorithmic potential as a technique for adaptation is their implicit parallelism. In a review by Hoffmeister (1991) various fine and coarse grain parallel implementations are presented, which are particularly well suited for MIMD–computers.

In the remainder of this paper we will focus only on the two lowest levels of the hierarchy, i.e. on Evolution Strategies (ESs) and Genetic Algorithms (GAs), which are described shortly in the following sections. The algorithms differ with respect to their learning capabilities. GAs are able to "learn", i.e. find, a point in the search space with maximum fitness. In order to do so they are driven by a set of fixed, external strategy parameters. This is referred to as *first level learning*. ESs work in a similar manner, but additionally they also "learn" the proper setting of several strategy parameters which guide the process of first level learning. This kind of *second level learning* is integrated into the algorithm, thus avoiding some kind of meta-level control algorithm like it was proposed by Grefenstette (1986) or Guliaev et al. (1989).

The effects of first and second level of genetic self-learning are demonstrated in the context of continuous parameter optimization which is used as an easy way to determine the "*fitness*" of evolving structures.

Genetic Algorithms

In this section only a very short introduction to Genetic Algorithms and their learning capability is given. For more informations on GAs the reader is referred to Holland (1975) and Goldberg (1989). The formal framework used here is taken from Hoffmeister and Bäck (1991) without going into the details. We focus on the following 8–tuple as an abstraction from Holland's work (1975):

$$\mathrm{GA} = (P^0, \lambda, l, s, \rho, \Omega, f, T) \qquad (1)$$

where

$$P^0 = (a_1^0, \ldots, a_\lambda^0) \in I^\lambda \quad \text{initial population}$$
$$I = \{0,1\}^l$$

[1]The Pittsburgh-approach treats a set of rules as one individual as described by Smith (1984) and Wilson and Goldbert (1989).

[2]The Michigan-approach treats one rule as one individual according to Holland and Reitman (1978).

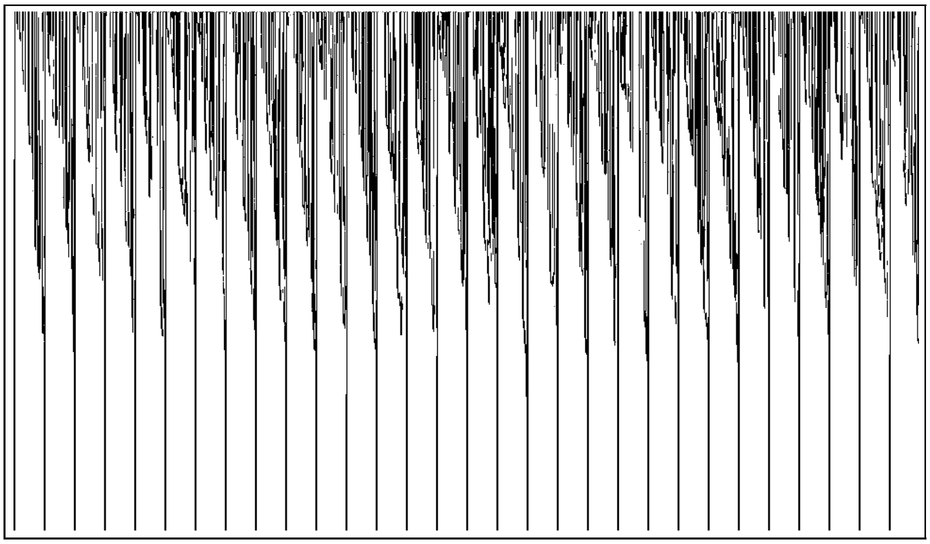

Figure 1: Development of best structure by a GA (first level learning)

λ	$\in$	$\mathbb{N}$	population size
l	$\in$	$\mathbb{N}$	bitlength of individuals
s	$:$	$I^\lambda \rightarrow I^\lambda$	selection operator
ρ	$:$	$I \rightarrow \Omega$	operator determination
Ω	$\subseteq$	$\{\omega : I \times I^\lambda \rightarrow I\}$	operator set
f	$:$	$I \rightarrow \mathbb{R}$	fitness function
T	$:$	$I^\lambda \rightarrow \{0,1\}$	termination criterion

P^0 is the randomly generated initial population, and the parameters λ and l describe the number of individuals representing one generation and the length of the 'genetic' representation of each individual, respectively.

The selection operator s produces an intermediate population P'^t from the population P^t by the generation of copies of elements from P^t: $P'^t = s(P^t)$. This is done by taking λ subsequent samples from $P^t = (a_1^t, \ldots, a_\lambda^t)$ according to the probability distribution $p_s : I \rightarrow [0,1]$, where $p_s(a_i^t) = f(a_i^t) \;/\; \sum_{j=1}^{\lambda} f(a_j^t)$ denotes the probability of individual a_i^t to be sampled. Holland (1975) introduced the name *proportional selection* for this selection scheme.

After the selection phase has taken place genetic operators are to be applied. One may think of a mapping $\rho : I \rightarrow \Omega$ which determines an operator $\omega_i^t \in \Omega$ for each individual $a_i'^t \in P'^t$ which will be applied to this individual: $\rho(a_i'^t) = \omega_i^t \; \forall i \in \{1, \ldots, \lambda\}$.

The genetic operator set $\Omega \subseteq \{\omega : I \times I^\lambda \rightarrow I\}$ includes genetic operators like *crossover* and *mutation*. The stochastic elements of these operators (application probabilities, e.g. $p_m \approx 0.001$ and $p_c \approx 0.6$, selection of loci) are hidden in the operators, e.g. application of an operator ω to an individual a_i and a population P yields a probability distribution $p : I \rightarrow [0,1]$ on I ($\sum_{a \in I} p(a) = 1$). According to p, a sample is taken from I, thus determining the resulting new individual $a_i' = \omega(a_i, P)$.

Now the transition from generation t to generation $t+1$ may be described as follows:

$$\begin{array}{rcll} s(P^t) & = & P'^t = (a_1'^t, \ldots, a_\lambda'^t) & \\ \rho(a_i'^t) & = & \omega_i^t & \forall i \in \{1, \ldots, \lambda\} \\ a_i^{t+1} & = & \omega_i^t(a_i'^t, P^t) & \forall i \in \{1, \ldots, \lambda\} \\ P^{t+1} & = & (a_1^{t+1}, \ldots, a_\lambda^{t+1}) & \end{array} \qquad (2)$$

The fitness values are obtained by the fitness function $f : I \rightarrow \mathbb{R}$, which is usually seen as a black box containing components like decoding of individuals, constraint handling, scaling and the objective function itself. $T : I^\lambda \rightarrow \{0,1\}$ denotes the termination criterion.

In order to illustrate the adaptive search capacity of such an algorithm, i.e. its first level learning properties, a simple optimization problem is used here. The fitness function f is a quadratic one of the form

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i^2 \qquad (3)$$

(sphere model). It is unimodal, i.e. only one minimum point $x^* = (0, \ldots, 0)$ exists. The task of the genetic algorithm is to locate the minimum point from a randomly initialized start population. The algorithm works by encoding each object variable $x_i \in \mathbb{R}$

by using 32 bits and a dimension of $n = 30$, such that $l = 960$. For decoding purposes, each object variable is restricted to the range $-5.12 \leq x_i \leq 5.12$ (the $2^{32}-1$ different integer numbers encoded by each substring of length 32 are linearly mapped to that interval of real values). In the special experiment described here a mutation rate $p_m = 0.001$, population size $\lambda = 50$, and a two–point crossover operator was used ($p_c = 0.6$). Furthermore, the algorithm uses a modified selection mechanism called *extinctive* selection, which was introduced to GAs by Bäck and Hoffmeister (1991a). In fact, the experiment described here is based on (10,50)-proportional selection.

The result is given in an uncommon graphical way in figure 1. Here for 4000 generations in steps of 5 generations the actually best individual of the population is given as a line of pixels, where a 1 corresponds to a black pixel and a 0 corresponds to a white one. The development towards the optimum string

$$\{1\underbrace{0\ldots0}_{31}\}^{30}$$

can be seen clearly in this illustration, without stressing too much numerical data (the final best objective function value is $4.26 \cdot 10^{-17}$). The figure demonstrates the capability of adaptation on the population level. The capability of adaptation on the meta–level, i.e. the level of strategy parameters, is demonstrated in the next section.

Evolution Strategies

Evolution strategies as devised by Schwefel (1977, 1981) are a class of methods primarily used for constrained and unconstrained parameter optimization of an objective function $f : \mathbb{R}^n \to \mathbb{R}$ and for problems with additional inequality constraints $g_l \geq 0$. The overall goal is to find a vector $x^\star \in \mathbb{R}^n$ such that $\forall x \in \mathbb{R}^n : f(x^\star) \leq f(x)$ in case of a minimization task. $x^\star$ is called the *global optimum*. Restriction to function minimization is without loss of generality since $\max f(x) = -\min(-f(x))$.

In general, non-linear objective functions exhibit multiple optima. A *local optimum* $\hat{x}$ is defined as

$$\begin{aligned} &\exists \epsilon > 0\ \forall x \in \mathbb{R}^n : \\ &\|x - \hat{x}\| < \epsilon \quad \Longrightarrow \quad f(\hat{x}) \leq f(x) \end{aligned} \tag{4}$$

Even if there is only one local optimum, it may be difficult to find a path to it in case of discontinuities. Trying to guarantee global convergence leads to more or less exhaustive scanning of the parameter space, since a lot of problems are NP-complete as demonstrated by Garey and Johnson (1979). Heuristic methods like the ES (or GA) try to keep within polynomial time without trapping into the nearest local optimum. They do not rely on gradient information like other search methods.

An ES incorporates the principles of population, recombination, mutation and selection from organic evolution as the major heuristics. Although it has very much in common with traditional gradient methods, which show just local convergence to some $\hat{x}$, it has a good chance to find better local minima due to the use of normally distributed mutations and sexual recombination. Schwefel's ES may be described as a 11-tupel

$$\mathrm{ES} = \left(P^0, \mu, \lambda,\ r, m, s,\ \Delta\sigma_g, \Delta\sigma, \Delta\theta,\ f, g, T\right) \tag{5}$$

where

P^0	$=$	$(a_1^0, \ldots, a_\mu^0) \in I^\mu$	population
			$I = \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^w$
μ	$\in$	$\mathbb{N}$	number of parents
λ	$\in$	$\mathbb{N}$	number of offspring
			$\lambda > \mu$
r	$:$	$I^\mu \to I$	recombination operator
m	$:$	$I \to I$	mutation operator
s	$:$	$I^\lambda \to I^\mu$	selection operator
$\Delta\sigma$	$\in$	$\mathbb{R}$	step-size meta-control
$\Delta\sigma_g$	$\in$	$\mathbb{R}$	step-size meta-control
$\Delta\theta$	$\in$	$\mathbb{R}$	correlation meta-control
f	$:$	$\mathbb{R}^n \to \mathbb{R}$	objective function
g_j	$:$	$\mathbb{R}^n \to \mathbb{R}$	constraint functions
			$j \in \{1, \ldots, q\}$
T	$:$	$I^\mu \to \{0,1\}$	termination criterion

where P_0 denotes the initial population of μ parents which produce λ offspring per generation. $\Delta\sigma$ and $\Delta\theta$ are parameters which control the mutation of the object variables $x \in \mathbb{R}^n$ on a meta-level. The "genetic" information of an individual $a = (x, \sigma, \theta) \in I$ consists of three parts, namely the set of object variables $x \in \mathbb{R}^n$, the set of standard deviations $\sigma \in \mathbb{R}^n$ for mutation of the object variables x, and a vector of inclination angles $\theta \in \mathbb{R}^w$ ($w = n/2(n-1)$ which are used to control correlated mutations of the object variables x. The later two are also referred to as *strategy parameters*, since they control the effects of the mutation operator m for an individual.

An ES works similar to an abstract automaton running sequentially through a set of states (generations) until the termination criterion T holds. The

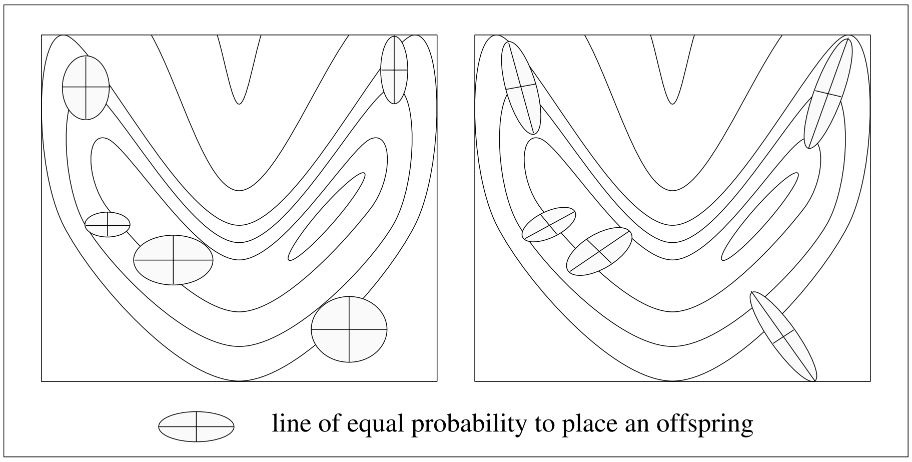

Figure 2: The effect of simple mutations (left) and correlated mutations (right)

term *generation* refers to the time step between successive states as well as to the population P_t at time t. The λ offspring are reduced to μ parents of the next generation by the selection operator s:

$$\begin{aligned} P^{t+1} &= s(P'^t) \\ P'^t &= (a_1'^t, \ldots, a_\lambda'^t) \in I^\lambda \\ a_j'^t &= m(r(P^t), \Delta\sigma_g, \Delta\sigma, \Delta\theta) \\ & \quad \forall g_j \in g : g_j(a_j'^t) \geq 0, \quad j \in \{1, \ldots, \lambda\} \end{aligned} \tag{6}$$

An offspring which does not satisfy all constraints g_j is simply ignored as a *lethal mutation*. When realizing the selection scheme "survival of the fittest" the selection operator s is defined such that

$$\forall a \in P^{t+1} \; \nexists a' \in P'^t : f(a') < f(a) \tag{7}$$

In the selection scheme presented here the life time of every individual in the population is restricted to one generation (pure selection). Such an ES is referred to as a (μ,λ)–ES, while a variant of it, which allows for an infinite survival of the parents by incorporating them into the selection, is called $(\mu+\lambda)$–ES (elitist strategy).

The recombination operator r is used to produce new offspring by mixing the information contained within different individuals of a population. Several recombination schemes may be envisaged:

$$r(P^t) = a' = (x', \sigma', \theta') \in I$$

$$x_i' = \begin{cases} x_{a,i} & \text{(A) no recombination} \\ x_{a,i} \text{ or } x_{b,i} & \text{(B) discrete} \\ \frac{1}{2}(x_{a,i} + x_{b,i}) & \text{(C) intermediate} \\ x_{a,i} \text{ or } x_{b_i,i} & \text{(D) global, discrete} \\ \frac{1}{2}(x_{a,i} + x_{b_i,i}) & \text{(E) global, intermediate} \end{cases} \tag{8}$$

where the indices $a, b, b_i \in P^t$ indicate parents likely chosen by r. Note, that with global recombination the mating partners for the recombination of a *single* component x_i' are chosen anew from the population resulting in a higher mixing of the genetic information than in the standard case (B).

Recombination is likewise defined for the other part of the genetic information, namely σ' and θ'. It realizes some kind of multi-point cross-over, which virtually enlarges the search space covered by the population.

The schemes (A), (B) and (C) may be found in nature when looking at non-sexual bacteria and sexually reproducing higher living beings, while (D) and (E) are somehow artificial.

The mutation operator m suffices to introduce new information into a population. It modifies at random the object *and* the strategy information of an individual $a = (x, \sigma, \theta)$ consisting of an object variable vector x, a vector of standard deviations σ, and a set of inclination angles θ, which influence the covariances of the mutations.

$$\begin{aligned} m(a) &= a' = (x', \sigma', \theta') \\ & \quad a, a' \in I; \; I = \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^w \\ \sigma_i' &= \sigma_i \cdot \exp\left(z_g + \mathbf{N_0}(\Delta\sigma)\right) \\ & \quad z_g = \mathbf{N_0}(\Delta\sigma_g) \\ \theta_j' &= \theta_j \cdot \exp \mathbf{N_0}(\Delta\theta) \quad ; \; j \in \{1, \ldots, w\} \\ x_i' &= x_i + \mathbf{N_0}(\mathbf{A}) \quad ; \; i \in \{1, \ldots, n\} \end{aligned} \tag{9}$$

where $\mathbf{N_0}(y)$ represents an independent, normally distributed random number with variance y^2 and expected value 0. $\mathbf{N_0}(\mathbf{A})$ is a normally distributed random vector z with expectation 0 and probability density

$$p(z) = \sqrt{\frac{\det \mathbf{A}}{(2\pi)^{\mathrm{n}}}}\ \exp\left(-\frac{1}{2} z^{\mathrm{T}} \mathbf{A}\, z\right) \tag{10}$$

The diagonal elements of the covariance matrix $\mathbf{A}^{-1}$ are the independent variances $\sigma_i'^2$ for the components x_i of the decision vector x, while the off-diagonal elements represent the covariances $c_{i,j}$ of the changes. Schwefel restricts the areas of equal probability density to n-dimensional hyperellipsoids, which are realized by a set of inclination angles $\theta' \in \mathbb{R}^w$, $w = n/2 \cdot (n-1)$ for the main axes of the hyperellipsoid. The standard deviations σ_i' serve as a kind of mean step size along those axes.

Note, that there are *two* parameters controlling the mutation of the step-sizes $\sigma \in \mathbb{R}^n$. Parameter $\Delta\sigma_g$ denotes a common scaling factor for all step-sizes, while $\Delta\sigma$ controls the individual variation of each step-size σ_i.

Introducing the strategy parameters σ and θ into the mutation–selection process is a major ingredience of an evolution strategy. It allows a population with its sets of strategy parameters to adapt to the *local* topology of the objective function dynamically. In general, good settings of the strategy parameters of an individual result in a better fitness according to f. Thus, selection automatically favours better settings. Schwefel (1987) demonstrated, that near-optimal step sizes are used for the direct search process of the evolution strategy resulting in a high overall rate of convergence. The effect of correlated mutations is illustrated by figure 2, where a set of equally likely mutations for an individual is sketched by the borderline of an ellipsoids. The length of the axes represent the setting of the different σ_i. With simple mutations the axes of an ellipsoid are parallel to the coordinate system, while in case of correlated mutations an ellipsoid may be freely oriented in space, thus exploiting narrow "valleys" in the search space much better.

The scheme presented realizes minimization of a single criteria based on haploid individuals, but it may be easily extended to a diploid or polyploid scheme. Kursawe (1990) presented an ES for multi-criteria optimization based on diploid individuals which not only finds a single point of the Pareto set but is able to mark the complete set within a single run.

Self-Learning

In order to demonstrate the feasibility of integrating the self-learning of strategy parameters like σ into an Evolutionary Algorithm, figure 3 shows some aspects of the dynamics inside an ES with simple mutations. Such an algorithm is a simple example for successful second level learning. Again, the optimization problem given by equation (3) is used. For demonstration purposes the global optimum point is shifted every 200 generations to show the self-learning capabilities in a nonstationary environment. The population size is also set to $\lambda = 50$ with $\mu = 8$.

Due to the symmetry of function f it is enough to learn just a single mutation rate (step-size), instead of 30 different ones. The left part of figure 3 shows the behaviour of an ES that has to learn only a single step-size, while the right part of the illustration shows an ES that must adapt 30 different step-sizes, i.e. an individual step-size for each object variable x_i.

The topmost plots show the observed first level learning, i.e. the way to the optimum expressed by the best, average and worst value of the objective function f per generation. The plots at the bottom of figure 3 present the maximal, average and minimal mutation rates (step-sizes) per generation which control the first level learning. First and second level learning are linked according to (9). Several interesting properties are reflected by figure 3.

The rate of convergence to the optimum is very much higher compared to GAs, which after 4000 generations find an $x \in \mathbb{R}^{30}$ such that $f(x) = 4.26 \cdot 10^{-17}$. An ES with simple mutations as used here yields an $f(x) \approx 10^{-8}$ after only 200 generations. Simulation runs with the undistorted function f show the same rate of convergence until the arithmetic resolution of the underlying computations is reached. This general advantage stems not only from the more adequate internal model used in ESs for this problem but *also* from the self-learning of strategy parameters which control the process of first level learning.

Theoretical considerations by Rechenberg (1973) and Schwefel (1977, 1981) for function f indicate that the optimal mutation rate is proportional to the distance from the optimum. A comparison of the top and bottom plots clearly shows that this proportion is maintained by the self-learned mutation rates, since $f(x)$ is nothing but the squared distance from the optimum and the shape of the curves agree perfectly. In general, ESs are able to learn fairly optimal settings of the mutation rates with respect to the requirements of the search space covered by the population which may change depending on location and time.

The rate of first and second level learning is also proportional to $1/n$, where n denotes the amount of self-learned information like the number of object variables and mutation rates. This relation results in the

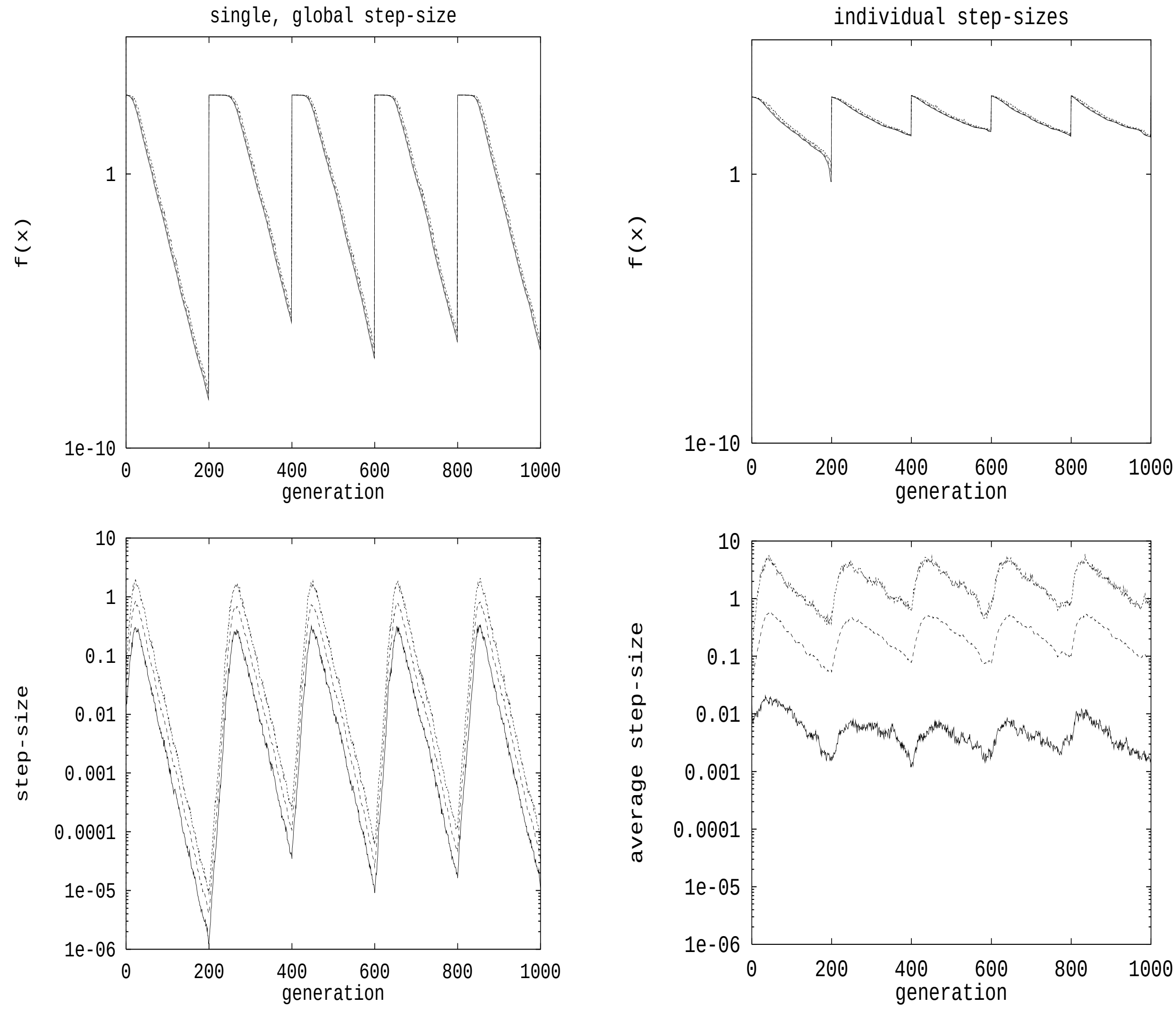

Figure 3: First and second level learning by an ES with different amounts of strategy parameters

different behaviour presented in the left and right part of figure 3. The more strategy parameters are to be learnt the more time is required for their self-learning. The much wider distribution of the 30 different mutation rates (right) instead of a just a single one (left) results from partially optimal settings of the object variables and σ-vectors, where several object variables and mutation rates are not adapted properly although the corresponding performance (fitness) of the individuals is among the μ best of a generation.

The plots at the bottom of figure 3 also show the speed at which the strategy parameters are adapted in case of a rapidly changing environment, i.e. shift of the optimum point. This rate is controlled by the external parameters $\Delta\sigma_g$ and $\Delta\sigma$, which also inversely depend on the amount of information to learn. The stagnation phase observed for the $f(x)$ after a shift of the optimum point directly depends on the time required to adjust the mutation rates to the proper magnitude. Then, mutation rates are scaled according to the current distance of the population from the (new) optimum point.

Conclusions

As demonstrated in the previous sections, learning in Evolutionary Algorithms is possible on two different levels: First, on the level of the population, i.e. the genetic information itself, and, more surprising, on the second level (meta–level) of the strategy parameters. The latter mechanism has two advantages: It improves the velocity of adaptation on the genotypic level by introducing the capability to learn an internal model of the topology of the objective function (step-sizes) and it simplifies the parameterization of the algorithm by removing parameters or at least by substituting many of them by fewer and less sensitive ones.

Finally, the self–learning of strategy parameters could lead to an adaptive algorithm without any externally predefined parameter, capable of self–adaptation of it's configuration and parameterization. These general principles are currently under investigation at the University of Dortmund, as well as the approach to use these mechanisms in Genetic Algorithms. This would allow for the application of parameter self–learning also to discrete optimization problems and could even help to solve some of the problems of learning and induction in Classifier-Systems, especially its problem of having too many sensitive external parameters.

Acknowledgements

Special thanks to Jörg Heitkötter for his engaged work on the graphics software which was used within the paper for the visualization of the results.

References

Alexander N. Antamoshkin, editor. *Random Search as a Method for Adaptation and Optimization of Complex Systems*, Divnogorsk, USSR, March 1991. Krasnojarsk Space Technology University.

Thomas Bäck and Frank Hoffmeister. Extended selection mechanisms in genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, pages 92–99, University of California, San Diego, USA, 1991. Morgan Kaufmann Publishers.

Thomas Bäck and Frank Hoffmeister. Global optimization by means of evolutionary algorithms. In Antamoshkin (1991), pages 17–21.

James Edward Baker. Adaptive selection methods for genetic algorithms. In Grefenstette (1985), pages 101–111.

James Edward Baker. Reducing bias and inefficiency in the selection algorithm. In J.J. Grefenstette, editor, *Proceedings of the second international conference on genetic algorithms and their applications*, pages 14–21, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, pages 235–282, 1989.

H. J. Bremermann, M. Rogson, and S. Salaff. Search by evolution. In M. Maxfield, A. Callahan, and L. J. Fogel, editors, *Biophysics and Cybernetic Systems — Proceedings of the 2nd Cybernetic Sciences Symposium*, pages 157–167, Los Angeles, CA, 1965. Spartan Books.

H. J. Bremermann. Numerical optimization procedures derived from biological evolution processes. In H. L. Oestreicher and D. R. Moore, editors, *Cybernetic Problems in Bionics*, pages 597–616. Gordon and Breach, New York, 1968.

M. L. Cramer. A representation for the adaptive generation of simple sequential programs. In Grefenstette (1985), pages 183–187.

Werner Ebeling and Reiner Feistel. *Physik der Selbstorganisation und Evolution.* Akademie–Verlag, Berlin, 1982.

Manfred Eigen. Selforganization of matter and the evolution of biological macromolecules. *Die Naturwissenschaften*, 58(10):465–523, October 1971.

L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, 1966.

C. Fujiko and J. Dickinson. Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 236–240, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the theory of NP–Completeness.* Freemann & Co., San Francisco, 1979.

David E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

J. J. Grefenstette, editor. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, New Jersey, 1985. Lawrence Erlbaum Associates.

John J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, SMC–16(1):122–128, 1986.

Yu. V. Guliaev, I. L. Bukatova, L. N. Golubeva, and V. F. Krapivin. Evolutionary informatics and "intelligent" special processors. Academy of Sciences of the USSR, Institute of Radio Engineering and Electronics, 1989. Preprint.

Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies: Similarities and differences. Technical Report "Grüne Reihe" No. 365, Department of Computer Science, University of Dortmund, November 1990.

Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In

Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 455–470. Springer, Berlin, 1991.

Frank Hoffmeister. Parallel evolutionary algorithms. In Antamoshkin (1991), pages 90–94.

John H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-directed inference systems*. Academic Press, New York, 1978.

John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. Classifier systems, Q–morphisms, and induction. In Lawrence Davis, editor, *Genetic algorithms and simulated annealing*, pages 116–128. Morgan Kaufmann Publishers, 1987.

John H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975.

John H. Holland. Escaping brittleness: The possibilities of general–purpose learning algorithms applied to parallel rule-based systems. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning Vol.II*, chapter 20, pages 593–623. Morgan Kaufmann Publishers, Inc., 1986.

John H. Holland. A mathematical framework for studying learning in classifier systems. *Physica*, 22D:307–317, 1986.

John H. Holland. Genetic algorithms and classifier systems: Foundations and future directions. In J. J. Grefenstette, editor, *Proceedings of the second international conference on genetic algorithms and their applications*, pages 82–89, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

John R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In N. S. Sridharan, editor, *Eleventh international joint conference on artificial intelligence*, pages 768–774. Morgan Kaufmann Publishers, August 1989.

John R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN–CS–90–1314, Department of Computer Science, Stanford University, Stanford, CA 94305, June 1990.

Frank Kursawe. *Evolutionsstrategien für die Vektoroptimierung*. Master thesis, University of Dortmund, Dortmund, Germany, February 1990.

Ryszard S. Michalski. Understanding the nature of learning: Issues and research directions. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning Vol.II*, chapter 1, pages 3–25. Morgan Kaufmann Publishers, Inc., 1986.

Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann–Holzboog Verlag, Stuttgart, 1973.

J. David Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, San Mateo, California, June 1989. Morgan Kaufmann Publishers.

Hans-Paul Schwefel. Binäre Optimierung durch somatische Mutation. Technical report, Technical University of Berlin and Medical University of Hannover, May 1975.

Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary systems research*. Birkhäuser, Basel, 1977.

Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.

Hans-Paul Schwefel. Collective phenomena in evolutionary systems. In *Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest*, volume 2, pages 1025–1033, June 1987.

Hans-Paul Schwefel. Evolutionary learning optimum–seeking on parallel computer architectures. In A. Sydow, S. G. Tzafestas, and R. Vichnevetsky, editors, *Proceedings of the International Symposium on Systems Analysis and Simulation 1988, I: Theory and Foundations*, pages 217–225. Akademie der Wissenschaften der DDR, Akademie-Verlag, Berlin, September 1988.

Hans-Paul Schwefel. Some observations about evolutionary optimization algorithms. In Hans-Michael Voigt, Heinz Mühlenbein, and Hans-Paul Schwefel, editors, *Evolution and Optimization '89*, Mathematical Ecology, pages 57–60, Berlin, 1989. Akademie Verlag Berlin.

S. F. Smith. Adaptive learning systems. In R. Forsyth, editor, *Expert systems — principles and case studies*, pages 168–189. Chapman and Hall, London, 1984.

Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank–based allocation of reproductive trials is best. In Schaffer (1989), pages 116–121.

Stewart W. Wilson and David E. Goldberg. A critical review of classifier systems. In Schaffer (1989), pages 244–255.