Self-Adaptation in Evolutionary Algorithms

Silja Meyer-Nieberg[1] and Hans-Georg Beyer[2]

[1] Department for Computer Science,
Universtät der Bundeswehr München,
D-85557 Neubiberg, Germany
silja.meyer-nieberg@unibw.de
[2] Research Center Process and Product Engineering,
Department of Computer Science,
Vorarlberg University of Applied Sciences,
Hochschulstr. 1, A-6850 Dornbirn, Austria
hans-georg.beyer@fhv.at

Summary. In this chapter, we will give an overview over self-adaptive methods in evolutionary algorithms. Self-adaptation in its purest meaning is a state-of-the-art method to adjust the setting of control parameters. It is called self-adaptive because the algorithm controls the setting of these parameters itself – embedding them into an individual's genome and evolving them. We will start with a short history of adaptation methods. The section is followed by a presentation of classification schemes for adaptation rules. Afterwards, we will review empirical and theoretical research of self-adaptation methods applied in genetic algorithms, evolutionary programming, and evolution strategies.

1 Introduction

Evolutionary algorithms (EA) operate on basis of populations of individuals. Their performance depends on the characteristics of the population's distribution. Self-Adaptation aims at biasing the distribution towards appropriate regions of the search space – maintaining sufficient diversity among individuals in order to enable further evolvability.

Generally, this is achieved by adjusting the setting of control parameters. Control parameters can be of various forms – ranging from mutation rates, recombination probabilities, and population size to selection operators (see e.g. [6]).

The goal is not only to find suitable adjustments but to do this efficiently. The task is further complicated by taking into account that the optimizer is faced by a dynamic problem since a parameter setting that was optimal at the beginning of an EA-run might become unsuitable during the evolutionary process. Thus, there is generally the need for a steady modification or adaptation of the control parameters during the run of an EA.

We will consider the principle of self-adaptation which is explicitly used in evolutionary programming (EP) [27, 26] and evolution strategies (ES) [49, 55] while it is rarely used in genetic algorithms (GA) [35, 36].

Individuals of a population have a set of object parameters that serves as a representative of possible solutions. The basic idea of explicit self-adaptation consists in incorporating the strategy parameters into the individual's genome and evolving them alongside with the object parameters.

In this paper, we will give an overview over the self-adaptive behavior of evolutionary algorithms. We will start with a short overview over the historical development of adaptation mechanisms in evolutionary computation. In the following part, i.e., Section 2.2, we will introduce classification schemes that are used to group the various approaches. Afterwards, self-adaptive mechanisms will be considered. The overview is started by some examples – introducing self-adaptation of the strategy parameter and of the crossover operator. Several authors have pointed out that the concept of self-adaptation may be extended. Section 3.2 is devoted to such ideas. The mechanism of self-adaptation has been examined in various areas in order to find answers to the question under which conditions self-adaptation works and when it could fail. In the remaining sections, therefore, we present a short overview over some of the research done in this field.

2 Adaptation and Self-Adaptation

2.1 A Short History of Adaptation in Evolutionary Algorithms

In this section, we will shortly review the historical development of adaptation mechanisms. The first proposals to adjust the control parameters of a computation automatically date back to the early days of evolutionary computation.

In 1967, Reed, Toombs, and Barricelli [51] experimented with the evolution of probabilistic strategies playing a simplified poker game. Half of a player's genome consisted of strategy parameters determining, e.g., the probabilities for mutation or the probabilities for crossover with other strategies. Interestingly, it was shown for a play for which a known optimal strategy existed that the evolutionary simulation realized nearly optimal plans.

Also in 1967, Rosenberg [52] proposed to adapt crossover probabilities. Concerning genetic algorithms, Bagley [9] considered incorporating the control parameters into the representation of an individual. Although Bagley's suggestion is one of the earliest proposals of applying classical self-adaptive methods, self-adaptation as usually used in ES appeared relatively late in genetic algorithms. In 1987, Schaffer and Morishima [54] introduced the self-adaptive *punctuated crossover* adapting the number and location of crossover points. Some years later, a first method to self-adapt the mutation operator was suggested by Bäck [7, 4]. He proposed a self-adaptive mutation rate in genetic algorithms similar to evolution strategies.

The idea of using a meta-GA can be found quite early. Here, an upper-level GA tries to tune the control parameters of a lower-level algorithm which tries in turn to solve the original problem. The first suggestion stems from Weinberg [71] giving rise to the work by Mercer and Sampson [45].

Concerning evolution strategies, the need to adapt the mutation strength (or strengths) appropriately during the evolutionary process was recognized in Rechenberg's seminal book *Evolutionsstrategie* [50].

He proposed the well-known 1/5th rule, which was originally developed for $(1+1)$-ES. It relies on counting the successful and unsuccessful mutations for a certain number of generations. If more than 1/5th of mutations leads to an improvement the mutation strength is increased and decreased otherwise. The aim was to stay in the so-called *evolution window* guaranteeing nearly optimal progress.

In addition to the 1/5th rule, Rechenberg [50] also proposed to couple the evolution of the strategy parameters with that of the object parameters. The strategy parameters were randomly changed. The idea of (explicit) self-adaptation was born. To compare the performance of this *learning population* with that of an ES using the 1/5th rule, Rechenberg conducted some experiments on the sphere and corridor model. The learning population exhibited a higher convergence speed and even more important it proved to be applicable in cases where it is improper to use the 1/5th rule. Self-adaptation thus appeared as a more universally usable method.

Since then various methods for adapting control parameters in evolutionary algorithms have been developed – ranging from adapting crossover probabilities in genetic algorithms to a direct adaptation of the distribution [16].

Schwefel [56, 58] introduced a self-adaptive method for changing the strategy parameters in evolution strategies which is today commonly associated with the term self-adaptation. In its most general form, the full covariance matrix of a general multidimensional normal distribution is adapted. A similar method of adapting the strategy parameters was offered by Fogel et al. [25] in the area of evolutionary programming – the so-called meta-EP operator for changing the mutation strength.

A more recent technique, the cumulative path-length control, stems from Ostermeier, Hansen, and Gawelczyk [48]. One of the aims is to derandomize the adaptation of the strategy parameters. The methods developed, the cumulative step-size adaptation (CSA) as well as the covariance matrix adaptation (CMA) [32], make use of an *evolution path*, $\mathbf{p}^{(g+1)} = (1-c)\mathbf{p}^{(g)} + \sqrt{c(2-c)}\mathbf{z}_{\text{sel}}^{(g+1)}$, which cumulates the selected mutation steps. To illustrate this concept, consider an evolution path where purely random selection is applied. Since the mutations are normally distributed, the cumulated evolution path is given by $\mathbf{u}^{(g)} = \sum_{k=1}^{g} \sigma\mathcal{N}^{(k)}(\mathbf{0},\mathbf{1})$, where $\mathcal{N}(\mathbf{0},\mathbf{1})$ is a random vector with identically independently distributed $\mathcal{N}(0,1)$ normally distributed components with zero mean and variance 1. The length of $u^{(g)}$ is χ-distributed with expectation $\overline{u} = \sigma\overline{\chi}$. Fitness based selection changes the situation. Too large mutation steps result in a selection of smaller mutations. Thus, the path-

length is smaller than $\overline{u}$ and the step size should be decreased. If on the other hand the path-length is larger than the expected $\overline{u}$, the step-size should be increased. CSA is also used in the CMA-algorithm. However, additionally the CMA adapts the whole covariance matrix [32] and as such it represents the state-of-the-art in real-coded evolutionary optimization algorithms.

2.2 A Taxonomy of Adaptation

As we have seen, various methods for changing and adapting control parameters of evolutionary algorithms exist and adaptation can take place on different levels.

Mainly, there are two taxonomy schemes [2, 22] which group adaptive computations into distinct classes – distinguishing by the type of adaptation, i.e., how the parameter is changed, and by the level of adaptation, i.e. where the changes occur. The classification scheme of Eiben, Hinterding, and Michaelewicz [22] extends and broadens the concepts introduced by Angeline in [2].

Let us start with Angeline's classification [2]. Considering the type of adaptation, adaptive evolutionary computations are divided into algorithms with *absolute update rules* and *empirical update rules*.

If an *absolute update rule* is applied, a statistic is computed. This may be done by sampling over several generations or by sampling the population. Based on the result, it is decided by means of a deterministic and fixed rule if and how the operator is to be changed. Rechenberg's 1/5th-rule [50] is one well-known example of this group.

In contrast to this, evolutionary algorithms with *empirical update rules* control the values of the strategy parameters themselves. The strategy parameter may be interpreted as an incorporated part of the individual's genome, thus being subject to "genetic variations". In the case the strategy parameter variation leads to an individual with a sufficiently good fitness, it is selected and "survives". Individuals with appropriate strategy parameters should – on average – have good fitness values and thus a higher chance of survival than those with badly tuned parameters. Thus, the EA should be able to self-control the parameter change.

As Smith [63] points out, the difference of the algorithms lies in the nature of the transition function. The transition function maps the set of parameters at generation t on that at $t+1$. In the case of absolute update rules, it is defined externally. In the case of self-adaptive algorithms, the transition function is a result of the operators and is defined by the algorithm itself.

Both classes of adaptive evolutionary algorithms can be further subdivided based on the level the adaptive parameters operate on. Angeline distinguished between population-, individual-, and component-level adaptive parameters.

Population-level adaptive parameters are changed globally for the whole population. Examples are for instance the mutation strength and the covariance matrix adaptation in CSA and CMA evolution strategies.

Adaptation on the individual-level changes the control parameters of an individual and these changes only affect that individual. The probability for crossover in GA is for instance adapted in [54] on the level of individuals.

Finally, component-level adaptive methods affect each component of an individual separately. Self-Adaptation in ES with correlated mutations (see Section 3.1) belongs to this type.

Angeline's classification was broadened by Eiben, Hinterding, and Michalewicz [22]. Adaptation schemes are again classified firstly by the type of adaptation and secondly – as in [2] – by the level of adaptation. Considering the different levels of adaptation a fourth level, *environment level adaptation*, was introduced to take into account cases where the responses of the environment are not static.

Concerning the adaptation type, in [22] the algorithms are first divided into *static*, i.e., no changes of the parameters occur, and *dynamic* algorithms. The term "dynamic adaptation" is used to classify any algorithm where the strategy parameters according to some rule, i.e., without external control. Based on the *mechanism of adaptation* three subclasses are distinguished: *deterministic*, *adaptive*, and finally *self-adaptive* algorithms. The latter comprise the same class of algorithms as in [2].

A deterministic adaptation is used if the control parameter is changed according to a deterministic rule *without* taking into account any present information by the evolutionary algorithm itself. Examples of this adaptation class are the time-dependent change of the mutation rates proposed by Holland [36] and the cooling schedule in simulated annealing like selection schemes.

Algorithms with an adaptive dynamic adaptation rule take feedback from the EA itself into account and change the control parameters accordingly. Again, a well known member of this class is Rechenberg's 1/5th-rule. Further examples (see [22])include Davis' adaptive operator fitness [15] and Julstrom's adaptive mechanism [38]. In both cases, the usage probability of an operator depends on its success or performance.

3 Self-Adaptation: The Principles

3.1 Self-Adapted Parameters: Some Examples

Self-Adaptation of Strategy Parameters

The technique most commonly associated with the term self-adaptation was introduced by Rechenberg [50] and Schwefel [56, 57] in the area of evolution strategies and independently by Fogel [24] for evolutionary programming. The control parameters considered here apply to the mutation process and parameterize the mutation distribution. The mutation is usually given by a normally distribution random vector, i.e. $\mathbf{Z} \sim \mathbf{N}(\mathbf{0}, \mathbf{C})$. The entries c_{ij} of the covariance matrix $\mathbf{C}$ are given by $c_{ii} = var(Z_i)$ or by $c_{ij} = cov(Z_i, Z_j)$ if $j \neq i$. The density function reads

$$p_Z(Z_1, \ldots, Z_N) = \frac{\mathrm{e}^{-\frac{1}{2}\mathbf{Z}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{Z}}}{\sqrt{(2\pi)^N \det(\mathbf{C})}}, \tag{1}$$

where N is the dimensionality of the search space. The basic step in the self-adaptation mechanism consists of a mutation of the mutation parameters themselves. In contrast to the additive change of the object variables, the mutation of the mutation strength strengths (i.e., the standard deviations $\sqrt{c_{ii}}$ in (1)) is realized by a multiplication with a random variable. The resulting mutation parameters are then applied in the variation of the object parameters. It should be mentioned here that concerning evolution strategies, the concept of self-adaptation was originally developed for non-recombinative $(1, \lambda)$-ES. Later on it was transferred to multi-parent strategies.

Figure 1 illustrates the basic mechanism of a multi-parent $(\mu/\rho, \lambda)$-ES with σ-self-adaptation. At generation g the ES maintains a population of μ candidate solutions – with the strategy parameters used in their creation. Based on that parent population, λ offspring are created via variation. The variation process usually comprises recombination and mutation. For each offspring, ρ parents are chosen for the recombination. First, the strategy parameters are changed. The strategy parameters of the chosen ρ parents are recombined and the result is mutated afterwards. The change of the object parameters occurs in the next step. Again, the parameters are first recombined and then mutated. In the mutation process, the newly created strategy parameter is used. After that, the fitness of the offspring is calculated. Finally, the μ-best individuals are chosen according to their fitness values as the next parental population. Two selection schemes are generally distinguished: "comma" and "plus"-selection. In the former case, only the offspring population is considered. In the latter, the new parent population is chosen from the old parent population and the offspring population.

Depending on the form of $\mathbf{C}$, different mutation distributions have to be taken into account. Considering the simplest case $\mathbf{Z} = \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$, the mutation of σ is given by

$$\sigma' = \sigma \mathrm{e}^{\tau\epsilon} \tag{2}$$

and using the new σ', the mutation of the object parameters reads

$$x_i' = x_i + \sigma'\mathcal{N}(0, 1). \tag{3}$$

The ϵ in Eq. (2) is a random number, often chosen as

$$\epsilon \sim \mathcal{N}(0, 1), \tag{4}$$

thus, producing log-normally distributed σ' variants. This way of choosing ϵ is also referred to as the "log-normal mutation rule". Equation (2) contains a new strategy specific parameter – the *learning rate* τ to be fixed. The general recommendation is to choose $\tau \propto 1/\sqrt{N}$, which has been shown to be optimal with respect to the convergence speed on the sphere [11].

BEGIN
  g:=0
  INITIALIZATION$\Big(\mathcal{P}_\mu^{(0)} := \Big\{\Big(\mathbf{y}_m^{(0)}, \sigma_m^{(0)}, F(\mathbf{y}_m^{(0)})\Big)\Big\}\Big)$
  REPEAT
    FOR EACH OF THE λ OFFSPRING DO
      $\mathcal{P}_\rho$:=REPRODUCTION($\mathcal{P}_\mu^{(g)}$)
      σ_l' :=RECOMB$_\sigma(\mathcal{P}_\rho)$;
      σ_l :=MUTATE$_\sigma(\sigma_r')$;
      $\mathbf{y}_l'$:=RECOMB$_\mathbf{y}(\mathcal{P}_\rho)$;
      $\mathbf{y}_l$:=MUTATE$_\mathbf{y}(\mathbf{x}_l', \sigma_l)$;
      $F_l := F(\mathbf{y}_l)$;
    END
    $\mathcal{P}_\lambda^{(g)}$:=$\Big\{(\mathbf{y}_l, \sigma_l, F_l)\Big\}$
    CASE ","-SELECTION: $\mathcal{P}_\mu^{(g+1)}$:=SELECT($\mathcal{P}_\lambda^{(g)}$)
    CASE "+"-SELECTION: $\mathcal{P}_\mu^{(g+1)}$:=SELECT($\mathcal{P}_\mu^{(g+1)}$,$\mathcal{P}_\lambda^{(g)}$)
    g:=g+1
  UNTIL stop;
END

Fig. 1. The $(\mu/\rho, \lambda)$-σSA-ES.

If different mutation strengths are used for each dimension, i.e., $Z_i = \sigma_i\mathcal{N}(0,1)$, the update rule

$$\sigma_i' = \sigma_i \exp\big(\tau'\mathcal{N}(0,1) + \tau\mathcal{N}_i(0,1)\big) \tag{5}$$
$$x_i' = x_i + \sigma_i'\mathcal{N}(0,1) \tag{6}$$

has been proposed. It is recommended [57] to choose the learning rates $\tau' \propto 1/\sqrt{2N}$ and $\tau \propto 1/\sqrt{2\sqrt{N}}$. The approach can also be extended to allow for correlated mutations [6]. Here, rotation angles α_i need to be taken into account leading to the update rule

$$\sigma_i' = \sigma_i \exp\big(\tau'\mathcal{N}(0,1) + \tau\mathcal{N}_i(0,1)\big) \tag{7}$$
$$\alpha_i' = \alpha_i + \beta\mathcal{N}_i(0,1) \tag{8}$$
$$\mathbf{x}' = \mathbf{x} + \mathcal{N}\big(0, \mathbf{C}(\sigma', \alpha)\big) \tag{9}$$

where $\mathbf{C}$ is the covariance matrix [6]. The parameter β is usually [6] chosen as 0.0873.

In EP [24], a different mutation operator, called *meta-EP*, is used

$$\sigma_i' = \sigma_i\Big(1 + \alpha\mathcal{N}(0,1)\Big) \tag{10}$$
$$x_i' = x_i + \sigma_i'\mathcal{N}(0,1). \tag{11}$$

Both operators lead to similar results – provided that the parameters τ and α are sufficiently small.

The log-normal operator, Eqs. (2), (3), and the meta-EP operator introduced above are not the only possibilities. Self-Adaptation seems to be relatively robust as to the choice of the distribution. Another possible operator is given by $\epsilon = \pm\delta$, where $+|\delta|$ and $-|\delta|$ are generated with the same probability of 1/2. That is, the resulting probability distribution function (pdf) of δ is a two-point distribution giving rise to the so-called two-point rule. It is usually implemented using $\delta = 1/\tau \ln(1+\beta)$, thus, leading with (2) to

$$\sigma_i' = \begin{cases} \sigma_i(1+\beta), & \text{if } u \leq 0.5 \\ \sigma_i/(1+\beta), & \text{if } u > 0.5 \end{cases} \tag{12}$$

where u is a random variable uniformly distributed on $(0, 1]$.

Another choice was proposed by Yao and Liu [73]. They substituted the normal distribution of the meta-EP operator with a Cauchy-distribution. Their new algorithm, called *fast evolutionary programming*, performed well on a set of test functions and appeared to be preferable in the case of multimodal functions. The Cauchy-distribution is similar to the normal distribution but has a far heavier tail. Its moments are undefined. In [42], Lee and Yao proposed to use a Lévy-distribution. Based on results on a suite of test functions, they argued that using Lévy-distributions instead of normal distribution may lead to higher variations and a greater diversity. Compared to the Cauchy-distribution, the Lévy-distribution allows for a greater flexibility since the Cauchy-distribution appears as a special case of the Lévy-distribution.

Self-Adaptation of Recombination Operators

Crossover is traditionally regarded as the main search mechanism in genetic algorithm and most efforts to self-adapt the characteristics of this operator stem from this area.

Schaffer and Morishima [54] introduced *punctuated crossover* which adapts the positions where crossover occurs. An individual's genome is complemented with a bitstring encoding crossover points. A position in this crossover map is changed in the same manner as its counterpart in the original genome. Schaffer and Morishima reported that punctuated crossover performed better than one-point crossover. Spears [67] points out, however, that the improvement of the performance might not necessarily be due to self-adaptation but due to the difference between crossover with more than one crossover point and one-point crossover.

Spears [67] self-adapted the form of the crossover operator using an additional bit to decide whether two-point or uniform crossover should be used for creating the offspring. Again, it should be noted that Spears attributes the improved performance not to the self-adaptation process itself but rather to the increased diversity that is offered to the algorithm.

Smith and Fogarty [62] introduced the so-called LEGO-algorithm, a linkage evolving genetic algorithm. The objects which are adapted are *blocks*, i.e.

linked neighboring genes. Each gene has two additional bits which indicate whether it is linked to its left and right neighbors. Two neighboring genes are then called *linked* if the respective bits are set. Offspring are created via successive tournaments. The positions of an offspring are filled from left to right by a competition between parental blocks. The blocks have to be eligible, i.e., they have to start at the position currently considered. The fittest block is copied as a whole and then the process starts anew. More than two parents may contribute in the creation of an offspring. Mutation also extends to the bits indicating linked genes.

3.2 A Generalized Concept of Self-Adaptation

In [6], two key features of self-adaptation have been identified: Self-adaptation aims at biasing the population distribution to more appropriate regions of the search space by making use of an indirect link between good strategy values and good object variables. Furthermore, self-adaptation relies on a population's diversity. While the adaptation of the operator ensures a good convergence speed, the degree of diversity determines the convergence reliability. More generally speaking, self-adaptation controls the relationship between parent and offspring population, i.e., the transmission function (see e.g. Altenberg [1]). The control can be direct by manipulating control parameters in the genome or more implicit. In the following, we will see that self-adaptation of strategy parameters can be put into a broader context.

Igel and Toussaint [37] considered the effects of neutral genotype-phenotype mapping. They pointed out that neutral genome parts give an algorithm the ability to "vary the search space distribution independent of phenotypic variation". This may be regarded as one of the main benefits of neutrality. Neutrality induces a redundancy in the relationship between genotype-phenotype. But neutral parts can influence the *exploration distribution* and thus the population's search behavior. This use of neutrality is termed generalized self-adaptation. It also comprises the classical form of self-adaptation since the strategy parameters adapted in classical self-adaptation belong to the neutral part of the genome.

More formally, generalized self-adaptation is defined as "adaptation of the exploration distribution $P_{\mathcal{P}}^{(t)}$ by exploiting neutrality – i.e. independent of changing phenotypes in the population, of external control, and of changing the genotype-phenotype mapping" [37]. Igel and Toussaint showed additionally that neutrality cannot be seen generally as a disadvantage. Although the state space is increased, it might not necessarily lead to a significant degradation of the performance.

In [28], Glickman and Sycara referred to an *implicit self-adaptation* caused by a non-injective genotype-phenotype mapping. Again there are variations of the genome that do not alter the fitness value but influence the transmission function which induces a similar effect.

Beyer and Deb [18] pointed out that in well-designed real-coded GAs, the parent offspring transmission function is controlled by the characteristics of the parent population. Thus, the GA performs an implicit form of self-adaptation. In contrast to the explicit self-adaptation in ES, an individual's genome does not contain any control parameters. Deb and Beyer [20] examined the dynamic behavior of real-parameter genetic algorithm (RCGA) that apply simulated binary crossover (SBX) [17, 21]. In SBX, two parents x^1 and x^2 create two offspring y^1 and y^2 according to

$$\begin{aligned} y_i^1 &= 1/2\Big((1-\beta_i)x_i^1 + (1+\beta_i)x_i^2\Big) \\ y_i^2 &= 1/2\Big((1+\beta_i)x_i^1 + (1-\beta_i)x_i^2\Big). \end{aligned} \tag{13}$$

The random variable β has the density

$$p(\beta) = \begin{cases} 1/2(\eta+1)\beta^{\eta}, & \text{if } 0 \leq \beta \leq 1 \\ 1/2(\eta+1)\beta^{-\eta-2}, & \text{if } \beta > 1 \end{cases} . \tag{14}$$

The authors pointed out that these algorithms show self-adaptive behavior although an individual's genome does not contain any control parameters. Well-designed crossover operators create offspring depending on the difference in parent solutions. The spread of children solutions is in proportion to the spread of the parent solutions, and near parent solutions are more likely to be chosen as children solutions than solutions distant from parents [20]. Thus, the diversity in the parental population controls that of the offspring population. Self-adaptation in evolution strategies has similar properties. In both cases, offspring closer to the parents have a higher probability to be created than individuals further away. While the implicit self-adaptability of real-coded crossover operators is well understood today, it is interesting to point out that even the standard one- or k-point crossover operators operating on binary strings do have this property: Due to the mechanics of these operators, bit positions which are common in both parents are transferred to the offspring. However, the other positions are randomly filled. From this point of view, crossover can be seen as a *self-adaptive mutation operator*, which is in contrast to the building block hypothesis [29] usually offered to explain the working of crossover in binary GAs.

3.3 Demands on the Operators: Real-coded Algorithms

Let us start with rules [12, 11, 13] for the design of mutation operators that stem from analyzes of implementations and theoretical considerations in evolution strategies: *reachability*, *scalability*, and *unbiasedness*. They state that every finite state must be reachable, the mutation operator must be tunable in order to adapt to the fitness landscape (scalability), and it must not introduce a bias on the population. The latter is required to hold also for the recombination operator [12, 40]. The demand of unbiasedness becomes clear when

considering that the behavior of an EA can be divided into two phases: Exploitation of the search space by selecting good solutions (reproduction) and exploration of the search space by means of variation. Only the former generally makes use of fitness information, whereas ideally the latter should only rely on search space information of the population. Thus, under a variation operator, the expected population mean should remain unchanged, i.e., the variation operators should not bias the population. This requirement, firstly made explicit in [12] may be regarded as a basic design principle for variation operators in EAs. The basic work [12] additionally proposed design principles with respect to the changing behavior of the population variance. Generally, selection changes the population variance. In order to avoid premature convergence, the variation operator must counteract that effect of the reproduction phase to some extend. General rules how to do this are, of course, nearly impossible to give but some *minimal* requirements can be proposed concerning the behavior on certain fitness landscapes [12].

For instance, Deb and Beyer [12] postulated that the population variance should increase exponentially with the generation number on flat or linear fitness functions. As pointed out by Hansen [31] this demand might not be sufficient. He proposed a linear increase of the expectation of the logarithm of the variance. Being based on the desired behavior in flat fitness landscapes, Beyer and Deb [12] advocate applying variation operators that increase the population variance also in the general case of unimodal fitness functions. While the variance should be decreased if the population brackets the optimum, this should not be done by the variation operator. Instead, this task should be left to the selection operator.

In the case of crossover operators in real-coded genetic algorithms (RCGA), similar guidelines have been proposed by Kita and Yamamura [40]. They suppose that the distribution of the parent population indicates an appropriate region for further search. As before, the first guideline states that the statistics of the population should not be changed. This applies here to the mean as well as to the variance-covariance matrix. Additionally, the crossover operator should lead to as much diversity in the offspring population as possible. It is noted that the first guideline may be violated since the selection operator typically reduces the variance. Therefore, it might be necessary to increase the present search region.

4 Self-Adaptation in EAs: Theoretical and Empirical Results

4.1 Genetic Algorithms

In this section, we will review the empirical and theoretical research that has been done in order to understand the working of self-adaptive EAs and to evaluate their performance.

Self-Adaptation of the Crossover Operator

Real-Coded GAs in Flat Fitness Landscapes

Beyer and Deb [19] analyzed three crossover operators commonly used in real-coded GAs, i.e., the simulated binary crossover (SBX) by Deb and Agrawala [17], the blend crossover operator (BLX) of Eshelman and Schaffer [23], and the *fuzzy recombination* of Voigt et al.[70].

In [19], expressions for the mean and the variance of the offspring population in relation to the parent population are derived. The aim is to examine if and under which conditions the postulates proposed in Section 3.3 are fulfilled. The fitness environments considered are flat fitness landscapes and the sphere. As mentioned before in Section 3.3, the self-adaptation should not change the population mean in the search space, i.e., it should not introduce a bias, but it should – since a flat fitness function is considered – increase the population variance and this exponentially fast.

It was shown in [19] that the crossover operator leaves the population mean unchanged regardless of the distribution of the random variable. Concerning the population variance, an exponential change can be asserted. Whether the variance expands or contracts depends on the population size and on the second moment of the random variable. Thus, a relationship between the population size and the distribution parameters of the random variables can be derived which ensures an expanding population.

Kita [39] investigated real-coded genetic algorithms using UNDX-crossover (unimodal normal distribution) and performed a comparison with evolution strategies. Based on empirical results, he pointed out that both appear to work reasonably well although naturally some differences in their behavior can be observed. The ES for example widens the search space faster when the system is far away from an optimum. On the other hand, the RCGA appears to have a computational advantage in high-dimensional search spaces compared to an ES which adapts the rotation angles of the covariance matrix according to Eqs. (7)-(9). Kita used a $(15, 100)$-ES with the usual recommendations for setting the learning rates.

Self-Adaptation of the Mutation Rate in Genetic Algorithms

Traditionally, the crossover (recombination) operator is regarded as the main variation operator in genetic algorithms, whereas the mutation operator was originally proposed as a kind of "background operator" [36] endowing the algorithm with the potential ability to explore the whole search space. Actually, there are good reasons to consider this as a reasonable recommondation in GAs with genotype-phenotype mapping from $\mathbb{B}^\ell \to \mathbb{R}^n$. As has been shown in [12], standard crossover of the genotypes does not introduce a bias on the population mean in the phenotype space. Interestingly, this does *not* hold for bit-flip mutations. That is, mutations in the genotype space result in a biased

phenotypic population mean – thus violating the postulates formulated in [12]. On the other hand, over the course of the years it was observed that for GAs on (pseudo) boolean functions (i.e., the problem specific search space is the $\mathbb{B}^\ell$) the mutation operator might also be an important variation operator to explore the search space (see e.g. [68]). Additionally, it was found that the optimal mutation rate or mutation probability does not only depend on the function to be optimized but also on the search space dimensionality and the current state of the search (see e.g. [5]).

A mechanism to self-adapt the mutation rate was proposed by Bäck [4, 7] for GA using the standard ES approach. The mutation rate is encoded as a bit-string and becomes part of the individual's genome. As it is common practice, the mutation rate is mutated first which requires its decoding to $[0, 1]$. The decoded mutation rate is used to mutate the positions in the bit-string of the mutation rate itself. The mutated version of the mutation probability is then decoded again in order to be used in the mutation of the object variables.

Several investigations have been devoted to the mechanism of self-adaptation in genetic algorithms. Most of the work is concentrated on empirical studies which are directed to possible designs of mutation operators trying to identify potential benefits and drawbacks.

Bäck [7] investigated the asymptotic behavior of the encoded mutation rate by neglecting the effects of recombination and selection. The evolution of the mutation rate results in a Markov chain[3]. Zero is an absorbing state of this chain which shows the convergence of the simplified algorithm.

The author showed empirically that in the case of GA with an extinctive selection scheme[4] self-adaptation is not only possible but can be beneficial [7]. For the comparison, three high-dimensional test functions (two unimodal, one multimodal) were used.

In [4], a self-adaptive GA optimizing the bit-counting function was examined. Comparing its performance with a GA that applies an optimal deterministic schedule to tune the mutation strength, it was shown that the self-adaptive algorithm realizes nearly optimal mutation rates.

The encoding of the mutation rate as a bit-string was identified in [8] as an obstacle for the self-adaptation mechanism. To overcome this problem, Bäck and Schütz [8] extended the genome with a real-coded mutation rate $p \in]0, 1[$. Several requirements have to be fulfilled. The expected change of p should be zero and small changes should occur with a higher probability than large ones. Also, there is the symmetry requirement that a change by a factor c should have the same probability as by $1/c$. In [8], a logistic normal distribution with parameter γ was used. The algorithm was compared with a GA without any adaptation and with a GA that uses a deterministic time-dependent schedule. Two selection schemes were considered. The GA with

[3] A Markov chain is a stochastic process which possesses the Markov property, i.e., the future behavior depends on the present state but not on the past.

[4] A selection scheme is extinctive iff at least one individual is not selected (see [7]).

the deterministic schedule performed best on the test-problems chosen with the self-adaptive GA in second place. Unfortunately, the learning rate γ was found to have a high impact.

Considering the originally proposed algorithm [7], Smith [65] showed that prematurely reduced mutation rates can occur. He showed that this can be avoided by using a fixed learning rate for the bitwise mutation of the mutation rate.

In 1996, Smith and Fogarty [61] examined empirically a self-adaptive steady state $(\mu + 1)$-GA finding that self-adaptation may improve the performance of a GA. The mutation rate was encoded again as a bit-string and several encoding methods were applied. Additionally, the impact of crossover in combination with a self-adaptive mutation rate was investigated. The self-adaptive GA appeared to be relative robust with respect to changes of the encoding or crossover.

In [66], the authors examined the effect of self-adaptation when the crossover operator and the mutation rate are both simultaneously adapted. It appeared that at least on the fitness functions considered synergistic effects between the two variation operators come into play.

Trying to model the behavior of self-adaptive GAs, Smith [64] developed a model to predict the mean fitness of the population. In the model several simplifications are made. The mutation rate is only allowed to assume q different values. Because of this, Smith also introduced a new scheme for mutating the mutation rate. The probability of changing the mutation rate is given by $P_a = z(q-1)/q$, where z is the so-called *innovation rate*.

In [69], Stone and Smith compared a self-adaptive GA using the log-normal operator with a GA with discrete self-adaptation, i.e., a GA implementing the model proposed in [64]. To this end, they evaluated the performance of a self-adaptive GA with continuous self-adaptation and that of their model on a set of five test functions. Stone and Smith found that the GA with discrete self-adaptation behaves more reliably whereas the GA with continuous self-adaptation may show stagnation. They attributed this behavior to the effect that the mutation rate gives the probability of bitwise mutation. As a result, smaller differences between mutation strengths are lost and more or less the same amount of genes are changed. The variety the log-normal operator provides cannot effectively be carried over to the genome and the likelihood of large changes is small. In addition, they argued that concerning the discrete self-adaptation a innovation rate of one is connected with an explorative behavior of the algorithm. This appears more suitable for multimodal problems whereas smaller innovation rates are preferable for unimodal functions.

4.2 Evolution Strategies and Evolutionary Programming

Research on self-adaptation in evolution strategies has a long tradition. The first theoretical in-depth analysis has been presented by Beyer [10]. It focused on the conditions under which a convergence of the self-adaptive algorithm

can be ensured. Furthermore, it also provided an estimate of the convergence order.

The evolutionary algorithm leads to a stochastic process which can be described by a Markov chain. The random variables chosen to describe the system's behavior are the object vector (or its distance to the optimizer, respectively) and the mutation strength.

There are several approaches to analyze the Markov chain. The first [14, 3] considers the chain directly whereas the second [59, 60, 34] analyzes induced supermartingales. The third [11, 18] uses a model of the Markov chain in order to determine the dynamic behavior.

Convergence Results using Markov Chains

Bienvenüe and François [14] examined the global convergence of adaptive and self-adaptive $(1, \lambda)$-evolution strategies on spherical functions. To this end, they investigated the induced stochastic process $z_t = \|x_t\|/\sigma_t$. The parameter σ_t denotes the mutation strength, whereas x_t stands for the object parameter vector.

They showed that (z_t) is a homogeneous Markov chain, i.e., z_t only depends on z_{t-1}. This also confirms an early result obtained in [10] that the evolution of the mutation strength can be decoupled from the evolution of $\|x_t\|$. Furthermore, they showed that (x_t) converges or diverges log-linearly – provided that the chain (z_t) is Harris-recurrent[5].

Auger [3] followed that line of research focusing on $(1, \lambda)$-ES optimizing the sphere model. She analyzed a general model of an $(1, \lambda)$-ES with

$$\begin{aligned} x_{t+1} &= \arg\min \left\{ f(x_t + \sigma_t \eta_t^1 \xi_t^1), \ldots, f(x_t + \sigma_t \eta_t^\lambda \xi_t^\lambda) \right\} \\ \sigma_{t+1} &= \sigma_t \eta^*(x_t),\ \eta^* \text{ given by } x_{t+1} = x_t + \sigma_t \eta^*(x_t) \xi^*(x_t), \end{aligned} \tag{15}$$

i.e., σ_{t+1} is the mutation strength which accompanies the best offspring. The function f is the sphere and η and ξ are random variables. Auger proved that the Markov chain given by $z_t = x_t/\sigma_t$ is Harris-recurrent and positive if some additional assumptions on the distributions are met and the offspring number λ is chosen appropriately. As a result, a law of large numbers can be applied and $1/t \ln(\|x_t\|)$ and $1/t \ln(\sigma_t)$ converge almost surely[6] to the same

[5] Let N_A be the number of passages in the set A. The set A is called Harris-recurrent if $P_z(N_A = \infty) = 1$ for $z \in A$. (Or in other words if the process starting from z visits A infinitely often with probability one.) A process (z_t) is Harris-recurrent if a measure ψ exists such that (z_t) is ψ-irreducible and for all A with $\psi(A) > 0$, A is Harris-recurrent (see e.g. [47]). A Markov process is called φ-irreducible if a measure φ exists so that for every set A with $\varphi(A) > 0$ and every state x holds: The return time probability to set A starting from x is greater than zero. A process is then called ψ-irreducible if it is φ-irreducible and ψ is a maximal irreduciblity measure fulfilling some additional propositions (see e.g. [47])

[6] A sequence of random variables x_t defined on the probability space $(\Omega, \mathcal{A}, P)$ converges almost surely to a random variable x if $P(\{\omega \in \Omega | \lim_{t\to\infty} x_t(\omega) =$

quantity – the convergence rate. This ensures either log-linearly convergence or divergence of the ES – depending on the sign of the limit. Auger further showed that the Markov chain (z_t) is also geometrically ergodic (see e.g. [47]) so that the Central Limit Theorem can be applied. As a result, it is possible to derive a confidence interval for the convergence rate. This is a necessary ingredient, because the analysis still relies on Monte-Carlo simulations in order to obtain the convergence rate (along with its confidence interval) numerically for the real $(1, \lambda)$-ES. In order to perform the analysis, it is required that the random variable ξ is symmetric and that both random variables ξ and η must be absolutely continuous with respect to the Lebesgue-measure. Furthermore, the density p_ξ is assumed to be continuous almost everywhere, $p_\xi \in L^\infty(\mathbb{R})$, and zero has to be in the interior of the support of the density, i.e., $0 \in \mathring{\text{supp}}\, p_\xi$. Additionally, it is assumed that $1 \in \mathring{\text{supp}}\, p_\eta$ and that $\mathrm{E}[|\ln(\eta)|] < \infty$ holds. The requirements above are met by the distribution functions normally used in practice, i.e., the log-normal distribution (mutation strength) and normal distribution (object variable). In order to show the Harris-recurrence, the positivity, and the geometric ergodicity, Forster-Lyapunov drift conditions (see e.g. [47]) need to be obtained. To this end, new random variables are to be introduced

$$\hat{\eta}(\lambda)\hat{\xi}(\lambda) = \min\left\{\eta^1\xi^1, \ldots, \eta^\lambda\xi^\lambda\right\}. \tag{16}$$

They denote the minimal change of the object variable. For the drift conditions a number α is required. Firstly, α has to ensure that the expectations $\mathrm{E}[|\xi|^\alpha]$ and $\mathrm{E}[(1/\eta)^\alpha]$ are finite. Provided that also $\mathrm{E}[|1/\hat{\eta}(\lambda)|^\alpha] < 1$, α can be used to give a drift condition V. More generally stated, α has to decrease the reduction velocity of the mutation strength associated with the best offspring of λ trials sufficiently. Thus, additional conditions concerning α and the offspring number λ are introduced leading to the definition of the sets

$$\mathit{\Gamma}_0 = \{\gamma > 0 : \mathrm{E}\left[|1/\eta|^\gamma\right] < \infty \text{ and } \mathrm{E}\left[|\xi|^\gamma\right] < \infty\} \tag{17}$$

and

$$\mathit{\Lambda} = \bigcup_{\alpha\in\mathit{\Gamma}_0} \mathit{\Lambda}_\alpha = \bigcup_{\alpha\in\mathit{\Gamma}_0} \{\lambda \in N : \mathrm{E}\left[1/\hat{\eta}(\lambda)^\alpha\right] < 1\}. \tag{18}$$

Finally, the almost sure convergence of $1/t\ln(\|x_t\|)$ and $1/t\ln(\sigma_t)$ can be shown for all $\lambda \in \mathit{\Lambda}$. It is not straightforward to give expression for $\mathit{\Lambda}$ or $\mathit{\Lambda}_\alpha$ in the general case although $\mathit{\Lambda}_\alpha$ can be numerically obtained for a given α. Only if the densities of η and ξ have bounded support, it can be shown that $\mathit{\Lambda}_\alpha$ is of the form $\mathit{\Lambda}_\alpha = \{\lambda : \lambda \geq \lambda_0\}$.

$x(\omega)\}) = 1$. Therefore, events for which the sequence does not converge have probability zero.

Convergence Theory with Supermartingales

Several authors [59, 60, 34] use the concept of martingales or supermartingales[7] to show the convergence of an ES or to give an estimate of the convergence velocity. As before, the random variables most authors are interested in are the object variable and the mutation strength.

Semenov [59] and Semenov and Terkel [60] examined the convergence and the convergence velocity of evolution strategies. To this end, they consider the stochastic Lyapunov function V_t of a stochastic process X_t. By showing the convergence of the Lyapunov function, the convergence of the original stochastic process follows under certain conditions.

From the viewpoint of probability theory, the function V_t may be regarded as a supermartingale. Therefore, a more general framework in terms of convergence of supermartingales can be developed. The analysis performed in [60] consists of two independent parts. The first concerns the conditions that imply almost surely convergence of supermartingales to a limit set. The second part (see also [59]) proposes demands on supermartingales which allow for an estimate of the convergence velocity. Indirectly, this also gives an independent convergence proof.

The adaptation of the general framework developed for supermartingales to the situation of evolution strategies requires the construction of an appropriate stochastic Lyapunov function. Because of the complicated nature of the underlying stochastic process, the authors did not succeed in the rigorous mathematical treatment of the stochastic process. Similar to the Harris-recurrent Markov chain approach, the authors had to resort to Monte-Carlo simulations in order to show that the necessary conditions are fulfilled.

In [59] and [60], $(1, \lambda)$-ES are considered where the offspring are generated according to

$$\begin{aligned} \sigma_{t,l} &= \sigma_t \mathrm{e}^{\vartheta_{t,l}} \\ x_{t,l} &= x_t + \sigma_{t,l}\zeta_{t,l} \end{aligned} \tag{19}$$

and the task is to optimize $f(x) = -|x|$. The random variables $\vartheta_{t,l}$ and $\zeta_{t,l}$ are uniformly distributed with $\vartheta_{t,l}$ assuming values in $[-2, 2]$ whereas $\zeta_{t,l}$ is defined on $[-1, 1]$. For this problem, it can be shown that the object variable and the mutation strength converge almost surely to zero – provided that there are at least three offspring. Additionally, the convergence velocity of the mutation strength and the distance to the optimizer is bounded from above by $\exp(-at)$ which holds asymptotically almost surely.

Hart, DeLaurentis, and Ferguson [34] also used supermartingales in their approach. They considered a simplified $(1, \lambda)$-ES where the mutations are modeled by discrete random variables. This applies to the mutations of the

[7] A random process X_t is called a supermartingale if $\mathrm{E}[|X_t|] < \infty$ and $\mathrm{E}[X_{t+1}|\mathcal{F}_t] \leq X_t$ where $\mathcal{F}_t$ is e.g. the σ-algebra that is induced by X_t.

object variables as well as to those of the mutation strengths. Offspring are generated according to

$$\begin{aligned} \sigma_{t,l} &= \sigma_t D \\ x_{t,l} &= x_t + \sigma_{t,l} B. \end{aligned} \tag{20}$$

The random variable D assumes three values $\{\gamma, 1, \eta\}$ with $\gamma < 1 < \eta$. The random variable B takes a value of either $+1$ or -1 with probability $1/2$ each. Under certain assumptions, the strategy converges almost surely to the minimum x^* of a function $f : \mathbb{R} \to \mathbb{R}$ which is assumed to be strictly monotonically increasing for $x > x^*$ and strictly monotonically decreasing for $x < x^*$.

As a second result, the authors proved that their algorithm fails to locate the global optimum of an example multimodal function with probability one. We will return to this aspect of their analysis in Section 5.

Instead of using a Lyapunov function as Semenov and Terkel did, they introduced a random variable that is derived from the (random) object variable and the mutation strength. It can be shown that this random variable is a nonnegative supermartingale if certain requirements are met. In that case, it can be shown that the ES converges almost surely to the optimal solution if the offspring number is sufficiently high.

The techniques introduced in [34] can be applied to the multi-dimensional case [33] provided that the fitness function is separable, i.e., $g(x) = \sum_{k=0}^{N} g_k(x_k)$ and the g_k fulfill the conditions for f. The authors considered an ES-variant where only one dimension is changed in each iteration. The dimension k is chosen uniformly at random. Let $X^t_{\lambda,k}$ and $\Sigma^t_{\lambda,k}$ denote the stochastic process that results from the algorithm. It can be shown that $X^t_{\lambda,1}, \ldots, X^t_{\lambda,N}$ are independent of each other. This also holds for $\Sigma^t_{\lambda,1}, \ldots, \Sigma^t_{\lambda,N}$. Therefore, the results of the one-dimensional analysis can be directly transferred.

Although the analysis in [34, 33] provides an interesting alternative, it is restricted to very special cases: Due to the kind of mutations used, the convergence results in [34, 33] are, however, not practically relevant if the number of offspring exceeds six.

Step-by-step Approach: The Evolution Equations

In 1996, Beyer [10] was the first to provide a theoretical framework for the analysis of self-adaptive EAs. He used approximate equations to describe the dynamics of self-adaptive evolution strategies. Let the random variable $r^{(g)} = \|X^{(g)} - \hat{X}\|$ denote the distance to the optimizer and $\varsigma^{(g)}$ the mutation strength. The dynamics of an ES can be interpreted as a Markov process as we have already seen. But generally, the transition kernels for

$$\begin{pmatrix} r^{(g)} \\ \varsigma^{(g)} \end{pmatrix} \to \begin{pmatrix} r^{(g+1)} \\ \varsigma^{(g+1)} \end{pmatrix} \tag{21}$$

cannot be analytically determined. One way to analyze the system is therefore to apply a step by step approach extracting the important features of the dynamic process and thus deriving approximate equations.

The change of the random variables can be divided into two parts. While the first denotes the expected change, the second covers the stochastic fluctuations

$$\begin{aligned} r^{(g+1)} &= r^{(g)} - \varphi(r^{(g)}, \varsigma^{(g)}) + \epsilon_R(r^{(g)}, \varsigma^{(g)}) \\ \varsigma^{(g+1)} &= \varsigma^{(g)}\Big(1 + \psi(r^{(g)}\varsigma^{(g)})\Big) + \epsilon_\sigma(r^{(g)}, \varsigma^{(g)}). \end{aligned} \tag{22}$$

The expected changes φ and ψ of the variables are termed *progress rate* if the distance is considered and *self-adaptation response* in the case of the mutation strength.

The distributions of the fluctuation terms ϵ_R and ϵ_σ are approximated using Gram-Charlier series', usually cut off after the first term. Thus, the stochastic term is approximated using a normal distribution. The variance can be derived considering the evolution equations. Therefore, the second moments have to be taken into account leading to the second-order progress rate and to the second-order self-adaptation response.

To analyze the self-adaptation behavior of the system, expressions for the respective progress rate and self-adaptation response have to be found. Generally, no closed analytical solution can be derived. Up to now, only results for $(1, 2)$-ES [11] using two-point mutations could be obtained. Therefore, several simplifications have to be introduced. For instance, if the log-normal operator is examined, the most important simplification is to consider $\tau \to 0$. The so derived expressions are then verified by experiments.

Self-Adaptation on the Sphere Model

It is shown in [11] that an $(1, \lambda)$-ES with self-adaptation converges to the optimum log-linearly. Also the usual recommendation of choosing the learning rate proportionally to $1/\sqrt{N}$, where N is the search space dimensionality, is indeed approximately optimal. In the case of $(1, \lambda)$-ES, the dependency of the progress on the learning rate is weak provided that $\tau \geq c/\sqrt{N}$ where c is a constant. As a result, it is not necessary to have N-dependent learning parameters.

As has been shown in [11], the time to adapt an ill-fitted mutation strength to the fitness landscape is proportionally to $1/\tau^2$. Adhering to the scaling rule $\tau \propto 1/\sqrt{N}$ results in an adaptation time that linearly increases with the search space dimensionality. Therefore, it is recommended to work with a *generation-dependent* or constant learning rate τ, respectively, if N is large.

The maximal progress rate that can be obtained in experiments is always smaller than the theoretical maximum predicted by the progress rate theory (without considering the stochastic process dynamics). The reason for this is that the fluctuations of the mutation strength degrade the performance.

The average progress rate is deteriorated by a loss part stemming from the variance of the strategy parameter. The theory developed in [11] is able to predict this effect qualitatively.

If recombination is introduced in the algorithm the behavior of the ES changes qualitatively. Beyer and Grünz [30] showed that multi-recombinative ES that use intermediate dominant recombination do not exhibit the same robustness with respect to the choice of the learning rate as $(1, \lambda)$-ES. Instead their progress in the stationary state has a clearly defined optimum. Nearly optimal progress is only attainable for a relatively narrow range of the learning rate τ. If the learning rate is chosen sub-optimally, the performance of the ES degrades but the ES still converges to the optimum with a log-linear rate. The reason for this behavior [46] is due to the different effects recombination has on the distance to the optimizer (i.e. on the progress rate) and on the mutation strength. An intermediate recombination of the object variables reduces the harmful parts of the mutation vector also referred to as "genetic repair effect". Thus, it reduces the loss part of the progress rate. This enables the algorithm to work with higher mutation strengths. However, since the strategy parameters are necessarily selected before recombination takes place, the self-adaptation response cannot reflect the after selection genetic repair effect and remains relatively inert to the effect of recombination.

Flat and Linear Fitness Landscapes

In [12], the behavior of multi-recombinative ES on flat and linear fitness landscapes was analyzed. Accepting the variance postulates proposed in [12] (see Section 3.3) the question arises whether the standard ES variation operators comply to these postulates, i.e., whether the strategies are able to increase the population variance in flat and linear fitness landscapes. Several common recombination operators and mutation operators were examined such as intermediate/dominant recombination of the object variables and intermediate/geometric recombination of the strategy parameters. The mutation rules applied for changing the mutation strength are the log-normal and the two-point distribution.

The analysis started with flat fitness landscapes which are selection neutral. Thus, the evolution of the mutation strength and the evolution of the object variables can be fully decoupled and the population variance can be easily computed. Beyer and Deb showed that if intermediate recombination is used for the object variables, the ES is generally able to increase the population variance exponentially fast. The same holds for dominant recombination. However, there is a memory of the old population variances that gradually vanishes. Whether this is a beneficial effect has not been investigated up to now.

In the case of linear fitness functions, only the behavior of $(1, \lambda)$-ES has been examined so far. It has been shown that the results obtained in [11] for the sphere model can be transferred to the linear case if $\sigma^* := \sigma N/R \to 0$ is

considered because the sphere degrades to a hyperplane. As a result, it can be shown that the expectation of the mutation strength increases exponentially if log-normal or two-point operators are used.

5 Problems and Limitations of Self-Adaptation

Most of the research done so far seems to be centered on the effects of self-adapting the mutation strengths. Some of the problems that were reported are related to divergence of the algorithm (see e.g. Kursawe [41]) and premature convergence. Premature convergence may occur if the mutation strength and the population variance are decreased too fast. This generally results in a convergence towards a suboptimal solution. While the problem is well-known, it appears that only a few theoretical investigations have been done. However, premature convergence is not a specific problem of self-adaptation. Rudolph [53] analyzed an $(1+1)$-ES applying Rechenberg's 1/5th-rule. He showed for a test problem that the ES's transition to the global optimum cannot be ensured when the ES starts at a local optimum and if the step-sizes are decreased to fast.

Stone and Smith [69] investigated the behavior of GA on multimodal functions applying Smith's discrete self-adaptation algorithm. Premature convergence was observed for low innovation rates and high selection pressure which causes a low diversity of the population. Diversity can be increased by using high innovation rates. Stone and Smith additionally argued that one copy of the present strategy parameter should be kept while different choices are still introduced. This provides a suitable relation between exploration and exploitation.

Liang et al. [43, 44] considered the problem of a prematurely reduced mutation strength. They started with an empirical investigation on the loss of step size control for EP on five test functions [43]. The EP used a population size of $\mu = 100$ and a tournament size of $q = 10$. Stagnation of the search occurred even for the sphere model. As they argued [43, 44], this might be due to the selection of an individual with a mutation strength far too small in one dimension but with a high fitness value. This individual propagates its ill-adapted mutation strength to all descendants and, therefore, the search stagnates.

In [44], Liang et al. examined the probability of loosing the step size control. To simplify the calculations, a $(1+1)$-EP was considered. Therefore, the mutation strength changes whenever a successful mutation happens. A loss of step size control occurs if after κ successful mutations the mutation strength is smaller than an arbitrarily small positive number ϵ. The probability of such an event can be computed. It depends on the initialization of the mutation strength, the learning parameter, on the number of successful mutations, and on ϵ. As the authors showed, the probability of loosing control of the step size increases with the number of successful mutations.

A reduction of the mutation strength should occur if the EP is already close to the optimum. However, if the reduction of the distance to the optimizer cannot keep pace with that of the mutation strength, the search stagnates. This raises the question whether the operators used in this EP implementation comply to the design principles postulated in [19] (compare Section 3.3 in this paper). An analysis of the EP behavior in flat or linear fitness landscapes might reveal the very reason for this failure. It should be also noted that similar premature convergence behaviors of self-adaptive ES are rarely observed. A way to circumvent such behavior is to introduce a lower bound for the step size. Fixed lower bounds are considered in [43]. While this surely prevents premature convergence of the EP, it does not take into account the fact that the ideal lower bound of the mutation strength depends on the actual state of the search.

In [44], two schemes are considered proposing a dynamic lower bound (DLB) of the mutation strength. The first is based on the success rate reminiscent of Rechenberg's 1/5th-rule. The lower bound is adapted on the population level. A high success rate leads to an increase of the lower bound, a small success decreases it. The second DLB-scheme is called "mutation step size based". For all dimensions, the average of the mutation strengths of the successful offspring is computed. The lower bound is then obtained as the median of the averages. These two schemes appear to work well on most fitness functions of the benchmark suite. On functions with many local optima, however, both methods exhibit difficulties.

As mentioned before, Hart, Delaurentis, and Ferguson [34] analyzed an evolutionary algorithm with discrete random variables on a multi-modal function. They showed the existence of a bimodal function for which the algorithm does not converge to the global optimizer with probability one if it starts close to the local optimal solution.

Won and Lee [72] addressed a similar problem although in contrast to Hart, DeLaurentis, and Ferguson they proved sufficient conditions for premature convergence avoidance of a $(1+1)$-ES on a one-dimensional bimodal function. The mutations are modeled using Cauchy-distributed random variables and the two-point operator is used for changing the mutation strengths themselves.

Glickman and Sycara [28] identified possible causes for premature reduction of the mutation strength. They investigated the evolutionary search behavior of an EA without any crossover on a complex problem arising from the training of neural networks with recurrent connections.

What they have called *bowl effect* may occur if the EA is close to a local minimum. Provided that the mutation strength is below a threshold then the EA is confined in a local attractor and it cannot find any better solution. As a result, small mutation strengths will be preferred.

A second cause is attributed to the selection strength. Glickman and Sycara suspect that if the selection strength is high, high mutation rates have a better chance of survival compared to using low selection strength: A large mutation rate increases the variance. This is usually connected with

a higher chance of degradation as compared to smaller mutation rates. But if an improvement occurs it is likely to be considerably larger than those achievable with small mutation rates. If only a small percentage of the offspring is accepted, there is a chance that larger mutation strengths "survive". Thus, using a high selection strength might be useful in safeguarding against premature stagnation. In their experiments, though, Glickman and Sycara could not observe a significant effect. They attributed this in part to the fact that the search is only effective for a narrow region of the selection strength.

Recently, Hansen [31] resumed the investigation of the self-adaptive behavior of multiparent evolution strategies on linear fitness functions started in [19]. Hansen's analysis is aimed at revealing the causes why self-adaptation usually works adequately on linear fitness functions. He offered conditions under which the control mechanism of self-adaptation fails, i.e., that the EA does not increase the step size as postulated in [19]. The expectation of the mutation strength is not measured directly. Instead, a function h is introduced the expectation of which is unbiased under the variation operators. The question that now remains to be answered is whether the selection will increase the expectation of $h(\sigma)$. In other words, is the effect of an increase of the expectation a consequence of selection (and therefore due to the link between good object vectors and good strategy values) or is it due to a bias introduced by the recombination/mutation-operators chosen?

Hansen proposed two properties an EA should fulfill: First, the descendants' object vectors should be point-symmetrically distributed after mutation and recombination. Additionally, the distribution of the strategy parameters given the object vectors after recombination and mutation has to be identical for all symmetry pairs around the point-symmetric center. Evolution strategies with intermediate multirecombination fulfill this symmetry assumption. Their descendants' distribution is point-symmetric around the recombination centroid.

Secondly, Hansen offered a so-called *σ-stationarity assumption*. It postulates the existence of a monotonically increasing function h whose expectation is left unbiased by recombination and mutation. Therefore, $\mathrm{E}[h(\mathcal{S}_k^{\sigma_{i;\lambda}|i=1,\ldots,\mu})] = 1/\mu \sum_{i=1}^{\mu} h(\sigma_{i;\lambda})$ must hold for all offspring. The term $\mathcal{S}_k^{\sigma_{i;\lambda}|i=1,\ldots,\mu}$ denotes the mutation strength of an offspring k created by recombination and mutation.

Hansen showed that if an EA fulfills the assumptions made above, self-adaptation does not change the expectation of $h(\sigma)$ if the offspring number is twice the number of parents.

The theoretical analysis was supplemented by an empirical investigation of the self-adaptation behavior of some evolution strategies examining the effect of several recombination schemes on the object variables and on the strategy parameter. It was shown that an ES which applies intermediate recombination to the object variables and to the mutation strength increases the expectation of $\log(\sigma)$ for all choices of the parent population size. On the other hand, evolution strategies that fulfill the symmetry and the stationarity assumption,

increase the expectation of $\log(\sigma)$ if $\lambda < \mu/2$, keep it constant for $\lambda = \mu/2$ and decrease it for $\lambda > \mu/2$.

Intermediate recombination of the mutation strengths results in an increase of the mutation strength. This is beneficial in the case of linear problems and usually works as desired in practice, but it might also have unexpected effects in other cases.

6 Outlook

Self-adaptation usually refers to an adaptation of control parameters which are incorporated into the individual's genome. These are subject to variation and selection - thus evolving together with the object parameters. Stating it more generally, a self-adaptive algorithm controls the transmission function between parent and offspring population by itself without any external control. Thus, the concept can be extended to include algorithms where the representation of an individual is augmented with genetic information that does not code information regarding the fitness but influences the transmission function instead.

Interestingly, real-coded genetic algorithms where the diversity of the parent population controls that of the offspring may be regarded as self-adaptive. Surprisingly, even binary GAs with crossover operators as 1-point or k-point crossover share self-adaptive properties to a certain extend.

Self-Adaptation is commonly used in the area of evolutionary programming and evolution strategies. Here, generally the mutation strength or the full covariance matrix is adapted. Analyses done so far focus mainly on the convergence to the optimal solution. Nearly all analyses use either a simplified model of the algorithm or have to resort to numerical calculations in their study. The results obtained are similar: On simple fitness functions considered, conditions can be derived that ensure the convergence of the EA to local optimal solutions. The convergence is usually log-linear.

The explicit use of self-adaptation techniques is rarely found in genetic algorithm and, if at all, mainly used to adopt the mutation rate. Most of the studies found are directed at finding suitable ways to introduce self-adaptive behavior in GA. As we have pointed out, however, crossover in binary standard GAs does provide a rudimentary form of self-adaptive behavior. Therefore, the mutation rate can be often kept at a low level provided that the population size is reasonably large. However, unlike the clear goals in real-coded search spaces, it is by no means obvious to formulate desired behaviors the self-adaptation should realize in binary search spaces. This is in contrast to some real-coded genetic algorithms where it can be shown mathematically that they can exhibit self-adaptive behavior in simple fitness landscapes.

It should be noted that self-adaptation techniques are not the means to solve all adaptation problems in EAs. Concerning evolution strategies, multi-recombinative self-adaptation strategies show sensitive behavior with respect

to the choices of the external learning rate τ. As a result, an optimal or a nearly optimal mutation strength is not always realized.

More problematic appear divergence or premature convergence to a suboptimal solution. The latter is attributed to a too fast reduction of the mutation strength. Several reasons for that behavior have been proposed although not rigorously investigated up to now. However, from our own research we have found that the main reason for a possible failure is due to the opportunistic way how self-adaptation uses the selection information obtained from just one generation. Self-adaptation rewards short-term gains. In its current form, it cannot look ahead. As a result, it may exhibit the convergence problems mentioned above.

Regardless of the problems mentioned, self-adaptation is a state-of-the-art adaptation technique with a high degree of robustness, especially in real-coded search spaces and in environments with uncertain or noisy fitness information. It also bears a large potential for further developments both in practical applications and in theoretical as well as empirical evolutionary computation research.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Center SFB 531 at the University of Dortmund and by the Research Center for Process- and Product-Engineering at the Vorarlberg University of Applied Sciences.

References

1. L. Altenberg. The evolution of evolvability in genetic programming. In K. Kinnear, editor, *Advances in Genetic Programming*, pages 47–74. MIT Press, Cambridge, MA, 1994.
2. P. J. Angeline. Adaptive and self-adaptive evolutionary computations. In M. Palaniswami and Y. Attikiouzel, editors, *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.
3. A. Auger. Convergence results for the $(1, \lambda)$-SA-ES using the theory of ϕ-irreducible Markov chains. *Theoretical Computer Science*, 334:35–69, 2005.
4. T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 85–94. North Holland, Amsterdam, 1992.
5. T. Bäck. Optimal mutation rates in genetic search. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8, San Mateo (CA), 1993. Morgan Kaufmann.
6. T. Bäck. Self-adaptation. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages C7.1:1–C7.1:15. Oxford University Press, New York, 1997.

7. Th. Bäck. Self-adaptation in genetic algorithms. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: proceedings of the first European conference on Artificial Life*, pages 263–271. MIT Press, 1992.
8. Th. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. In *ISMIS*, pages 158–167, 1996.
9. J. D. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, 1967.
10. H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1996.
11. H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heidelberg, 2001.
12. H.-G. Beyer and K. Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, 2001.
13. H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.
14. A. Bienvenüe and O. François. Global convergence for evolution strategies in spherical problems: Some simple proofs and d difficulties. *Theoretical Computer Science*, 308:269–289, 2003.
15. L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proc. 3rd Int'l Conf. on Genetic Algorithms*, pages 61–69, San Mateo, CA, 1989. Morgan Kaufmann.
16. M. W. Davis. The natural formation of gaussian mutation strategies in evolutionary programming. In *roceedings of the Third Annual Conference on Evolutionary Programming*, San Diego, CA, 1994. Evolutionary Programming Society.
17. K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.
18. K. Deb and H.-G. Beyer. Self-adaptation in real-parameter genetic algorithms with simulated binary crossover. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 172–179, San Francisco, CA, 1999. Morgan Kaufmann.
19. K. Deb and H.-G. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. Series CI 61/99, SFB 531, University of Dortmund, March 1999.
20. K. Deb and H.-G. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.
21. K. Deb and M. Goyal. A robust optimization procedure for mechanical component design based on genetic adaptive search. *Transactions of the ASME: Journal of Mechanical Desiogn*, 120(2):162–164, 1998.
22. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
23. L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.
24. D. B. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California, San Diego, 1992.
25. D. B. Fogel, L. J. Fogel, and J. W. Atma. Meta-evolutionary programming. In R.R. Chen, editor, *Proc. of 25th Asilomar Conference on Signals, Systems & Computers*, pages 540–545, Pacific Grove, CA, 1991.

26. L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.
27. L.J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.
28. M. Glickman and K. Sycara. Reasons for premature convergence of self-adapting mutation rates. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 62–69, Piscataway, NJ, 2000. IEEE Service Center.
29. D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, MA, 1989.
30. L. Grünz and H.-G. Beyer. Some observations on the interaction of recombination and self-adaptation in evolution strategies. In P.J. Angeline, editor, *Proceedings of the CEC'99 Conference*, pages 639–645, Piscataway, NJ, 1999. IEEE.
31. N. Hansen. Limitations of mutative σ-self-adaptation on linear fitness functions. *Evolutionary Computation*, 2005. accepted for publication.
32. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
33. W. E. Hart and J. M. DeLaurentis. Convergence of a discretized self-adapative evolutionary algorithm on multi-dimensional problems. submitted.
34. W.E. Hart, J.M. DeLaurentis, and L.A. Ferguson. On the convergence of an implicitly self-adaptive evolutionary algorithm on one-dimensional unimodal problems. *IEEE Transactions on Evolutionary Computation*, 2003. To appear.
35. J. H. Holland. Outline for a logical theory of adaptive systems. *JACM*, 9:297–314, 1962.
36. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.
37. Ch. Igel and M. Toussaint. Neutrality and self-adaptation. *Natural Computing: an international journal*, 2(2):117–132, 2003.
38. B. A. Julstrom. Adaptive operator probabilities in a genetic algorithm that applies three operators. In *SAC*, pages 233–238, 1997.
39. H. Kita. A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms. *Evolutionary Computation*, 9(2):223–241, 2001.
40. H. Kita and M. Yamamura. A functional specialization hypothesis for designing genetic algorithms. In *Proc. IEEE International Conference on Systems, Man, and Cybernetics '99*, pages 579–584, Piscataway, New Jersey, 1999. IEEE Press.
41. F. Kursawe. *Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien — Metastrategien*. Dissertation, Fachbereich Informatik, Universität Dortmund, 1999.
42. Ch.-Y. Lee and X. Yao. Evolutionary programming using mutations based on the levy probability distribution. *Evolutionary Computation, IEEE Transactions on*, 8(1):1– 13, Feb. 2004.
43. K.-H. Liang, X. Yao, Ch. N. Newton, and D. Hoffman. An experimental investigation of self-adaptation in evolutionary programming. In V. W. Porto, N. Saravanan, Waagen D. E., and A. E. Eiben, editors, *Evolutionary Programming*, volume 1447 of *Lecture Notes in Computer Science*, pages 291–300. Springer, 1998.
44. K.-H. Liang, X. Yao, and Ch. S. Newton. Adapting self-adaptive parameters in evolutionary algorithms. *Artificial Intelligence*, 15(3):171 – 180, November 2001.
45. R. E. Mercer and J. R. Sampson. Adaptive search using a reproductive metaplan. *Kybernetes*, 7:215–228, 1978.

46. S. Meyer-Nieberg and H.-G. Beyer. On the analysis of self-adaptive recombination strategies: First results. In B. McKay et al., editors, *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, UK*, pages 2341–2348, Piscataway NJ, 2005. IEEE Press.
47. S. P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer, 1993.
48. A. Ostermeier, A. Gawelczyk, and N. Hansen. A derandomized approach to self-adaptation of evolution strategies. *Evolutionary Computation*, 2(4):369–380, 1995.
49. I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment, Farnborough*, page Library Translation 1122, 1965.
50. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973.
51. J. Reed, R. Toombs, and N.A. Barricelli. Simulation of biological evolution and machine learning. i. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology*, 17:319–342, 1967.
52. R.S. Rosenberg. *Simulation of genetic populations with biochemical properties*. Ph.d. dissertation, Univ. Michigan, Ann Arbor, MI, 1967.
53. G. Rudolph. Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation*, 5(4):410–414, 2001.
54. J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J.J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proc. of the Second Int'l Conference on Genetic Algorithms*, pages 36–40, 1987.
55. H.-P. Schwefel. Kybernetische Evolution als Strategie der exprimentellen Forschung in der Strömungstechnik. Master's thesis, Technical University of Berlin, 1965.
56. H.-P. Schwefel. Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit. Technical report, Technical University of Berlin, 1974. Abschlußbericht zum DFG-Vorhaben Re 215/2.
57. H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Interdisciplinary systems research; 26. Birkhäuser, Basel, 1977.
58. H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.
59. M.A. Semenov. Convergence velocity of evolutionary algorithm with self-adaptation. In *GECCO 2002*, pages 210–213, 2002.
60. M.A. Semenov and D.A. Terkel. Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic lyapunov function. *Evolutionary Computation*, 11(4):363–379, 2003.
61. J. Smith and T. C. Fogarty. Self-adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC '96)*, pages 318–323. IEEE Press, NY, 1996.
62. J. Smith and T.C. Fogarty. Recombination strategy adaptation via evolution of gene linkage. In *Proc. of the 1996 IEEE International Conference on Evolutionary Computation*, pages 826–831. IEEE Publishers, 1996.
63. J. E. Smith. *Self-Adaptation in Evolutionary Algorithms*. PhD thesis, University of the West of England, Bristol, 1998.

64. J. E. Smith. Modelling gas with self adaptive mutation rates. In L. Spector and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 599–606, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
65. J. E. Smith. Parameter perturbation mechanisms in binary coded gas with self-adaptive mutation. In K. DeJong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms 7*, pages 329–346. Morgan Kauffman, 2004.
66. J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, June 1997.
67. W. Spears. Adapting crossover in evolutionary algorithms. In *Proceedings of the Evolutionary Programming Conference*, pages 367–384, 1995.
68. W.M. Spears. *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer-Verlag, Heidelberg, 2000.
69. Ch. Stone and J. E. Smith. Strategy parameter variety in self-adaptation of mutation rates. In W. B. Langdon and et al., editors, *GECCO*, pages 586–593. Morgan Kaufmann, 2002.
70. H.-M. Voigt, H. Mühlenbein, and D. Cvetković. Fuzzy recombination for the breeder genetic algorithm. In L. J. Eshelman, editor, *Proc. 6th Int'l Conf. on Genetic Algorithms*, pages 104–111, San Francisco, CA, 1995. Morgan Kaufmann Publishers, Inc.
71. R. Weinberg. *Computer Simulation of a Living Cell*. PhD thesis, University of Michigan, 1970.
72. J. M. Won and J. S. Lee. Premature convergence avoidance of self-adaptive evolution strategy,. In *The 5th International Conference on Simulated Evolution And Learning*, Busan, Korea, Oct 2004.
73. X. Yao and Y. Liu. Fast evolutionary programming. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 451–460. The MIT Press, Cambridge, MA, 1996.