Genetic Algorithms in Engineering and Computer Science

Genetic Algorithms in Engineering and Computer Science

Edited by J. Périaux and G. Winter

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

6

Evolution Strategies I: Variants and their computational implementation

Thomas Bäck[1] and Hans-Paul Schwefel[2]

6.1 INTRODUCTION

The first evolution strategy (ES) was developed in 1964 at the Technical University of Berlin (TUB) by Rechenberg and Schwefel as an experimental optimization technique. First applications of the strategy dealt with problems like driving a flexible pipe bending or changeable nozzle contour into a shape with minimal loss of energy [KS70]. This early variant of an evolution strategy, the so-called (1+1)-ES, works on the basis of two individuals only, i.e., one parent and one descendant per generation. The descendant is created by applying binomially distributed variations (with expectation zero and variance σ^2) to the parent, and either the descendant becomes parent of the next generation, if it is better than its parent, or the parent "survives." The (1+1)-ES was soon used also for continuous decision variables (with normally distributed variations), the latter presently being the major application domain. Rechenberg [Rec73] developed a theory of convergence velocity for the (1+1)-ES, and derived a rule for deterministically adapting the standard deviation of mutations according to the measured success frequency of mutations. According to an average of the optimal success probabilities for two simple model functions (corridor model and sphere model), the rule was called 1/5-success rule.

[1] Informatik Centrum Dortmund, Joseph-von-Fraunhofer-Str. 20, 44227 Dortmund, Germany. E-mail: baeck@ls11.informatik.uni-dortmund.de
[2] University of Dortmund, Department of Computer Science, 44221 Dortmund, Germany. E-mail: schwefel@ls11.informatik.uni-dortmund.de

Genetic Algorithms in Engineering and Computer Science
Editor J. Périaux and G. Winter

Soon, the (1+1)-ES was substituted, especially in computer applications, by variants with $\mu > 1$ parents and $\lambda > 1$ descendants per generation. Nevertheless, for historical and completeness reasons, the (1+1)-ES is discussed in section 6.2 of this overview.

The first multimembered evolution strategy, the (μ+1)-ES with $\mu > 1$, already introduced the concept of recombination to form one offspring individual from a mix of parental attributes. After mutation and fitness evaluation, the offspring replaces the worst parent individual if it performs better than that. Although it was never widely used, this strategy provided the basic idea to facilitate the transition to the modern ($\mu \overset{+}{,} \lambda$)-strategies, that represent the state-of-the-art [Rec94, Sch95]. In case of a (μ+λ)-strategy, μ parents create λ descendants by recombination and mutation, and the μ best individuals out of parents plus descendants are deterministically selected to become parents of the new generation. In case of a (μ,λ)-strategy, the μ best individuals out of the descendants only form the new parent set; in this case, $\lambda \geq \mu$ is required, otherwise $\lambda \geq 1$ is sufficient.

Besides a population of individuals and recombination of parental information, the (μ,λ)-ES substitutes the deterministic step-size control by a self-adaptation process that learns step sizes (and, optionally, covariances) on-line during the evolutionary search process [HB92, Sch95]. The ($\mu \overset{+}{,} \lambda$)-strategy is presented in detail in section 6.4 of this paper.

Ongoing research efforts led to the development of a variety of special variants of evolution strategies, e.g. for exploiting parallel hardware architectures or for solving multiple criteria decision making problems. A brief overview of these variants is given in section 6.5.

Finally, we conclude by outlining a proposed generalized evolution strategy which allows for a gradual transition between (μ+λ)-selection and (μ,λ)-selection by introducing a maximal life span κ of the individuals. The contemporary $(\mu, \kappa, \lambda, \rho)$-strategy also incorporates some other features which either theoretically or experimentally proved to be useful extensions of the algorithm [SR95].

6.1.1 Notation

Throughout this paper, we assume an n-dimensional, continuous parameter optimization problem of the form

$$f^* := f(\vec{x}^*) = \min\{f(\vec{x}) \mid \vec{x} \in M \subseteq \boldsymbol{R}^n\} \quad , \tag{6.1}$$

where $M = \{\vec{x} \in \boldsymbol{R}^n \mid g_j \geq 0 \; \forall j \in \{1, \ldots, q\}\}$ is the set of feasible points and $g_j : \boldsymbol{R}^n \to \boldsymbol{R}$ are inequality constraints. Individuals $\vec{a} \in I$ are elements of an individual space $I = \boldsymbol{R}^n \times S$, where the set S of strategy parameters depends on the kind of evolution strategy. $P^{(t)} = \{\vec{a}_1, \ldots, \vec{a}_k\} \in I^k$ denotes a population[1] of $k \in \{\mu, \lambda\}$ individuals at generation t, where a population is interpreted as a multiset of elements from I (duplicates might occur in the population). $\mu, \lambda \in N$ denote the number of parents and of descendants, respectively. The genetic operators are denoted by

[1] Because the cardinality of all sets is known in advance and elements are often considered in a specific order (by objective function values) and need to be indexed, we take the freedom to interchange vector and set notation by writing $\{\vec{a}_1, \ldots, \vec{a}_k\} \in I^k$ (rather than $\subset 2^I$).

mappings

$$\begin{array}{lcll} \mathbf{rec} & : & I^\mu \rightarrow I & \text{recombination} \\ \mathbf{mut} & : & I \rightarrow I & \text{mutation} \\ \mathbf{sel}_\mu^k & : & I^k \rightarrow I^\mu & \text{selection}, k \in \{\lambda, \mu+\lambda\} \end{array} \tag{6.2}$$

which also depend on additional, operator-specific parameters such as step length variabilities $\tau, \tau_0, \beta \in \boldsymbol{R}_+$ for mutation and recombination type $\omega \in \{0, 1, 2, 3\}$.

A single iteration of the evolution strategy, i.e., a step from a population $P^{(t)}$ towards the next parent population $P^{(t+1)}$ is modeled by the mapping

$$opt_{ES} : I^\mu \rightarrow I^\mu \tag{6.3}$$

according to

$$opt_{ES}(P^{(t)}) = \mathbf{sel}_\mu^k(\sqcup_{i=1}^\lambda \{\mathbf{mut}(\mathbf{rec}(P^{(t)}))\} \sqcup Q) \tag{6.4}$$

where $Q \in \{P^{(t)}, \emptyset\}$ depends on the selection operator ((μ,λ)-selection: $Q = \emptyset$, $k = \lambda$; (μ+λ)-selection: $Q = P^{(t)}$, $k = \mu + \lambda$), $\sqcup$ denotes the union operator on multisets, and $\mu = 1$ is supposed to imply no recombination, i.e., $\mathbf{rec} = \mathbf{id}$.

The notation $z \sim \mathbf{N}(\zeta, \sigma^2)$ denotes a realization of a normally distributed random variable with expectation ζ and variance σ^2, and $u \sim \mathbf{U}(\cdot)$ is a realization of a uniformly distributed random variable (the argument is either a continuous interval or a finite set of values). In contrast to uniformly distributed random numbers, most programming language libraries do not provide a function to generate normally distributed random numbers. From a pair $u_1, u_2 \sim \mathbf{U}((0, 1])$ of uniformly distributed random numbers, however, one can easily generate two independent, normally distributed random numbers with expectation zero and variance one according to (see [Sch95], pp. 115-116):

$$z_1 = \sqrt{-2 \ln u_1} \sin(2\pi u_2) \quad , \quad z_2 = \sqrt{-2 \ln u_1} \cos(2\pi u_2) \quad .$$

$z_1, z_2 \sim \mathbf{N}(0, 1)$ can be transformed by $z_i' = \sigma_i z_i$ to obtain random numbers distributed according to $\mathbf{N}(0, \sigma_i^2)$.

6.2 THE $(1+1)$-EVOLUTION STRATEGY

As outlined in the introduction, the (1+1)-ES is characterized by $\mu = \lambda = 1$, the absence of recombination, and a deterministic rule for the modification of the (one and only) step size σ for all mutations. An individual $\vec{a} = (\vec{x}, \sigma) \in \boldsymbol{R}^n \times \boldsymbol{R}_+$ consists of the object variable vector $\vec{x}$ and one standard deviation[2] σ. The mutation operator is a composition of the (deterministic) modification[3] $\mathbf{mu}_\sigma$ of σ and the mutation $\mathbf{mu}_x$ of $\vec{x}$:

$$\mathbf{mut} = \mathbf{mu}_x \circ \mathbf{mu}_\sigma \quad , \tag{6.5}$$

[2] We would like to add the remark that different step sizes $\sigma_i = \sigma \cdot s_i^{(0)}$ have already been used in the (1+1)-ES also, but the relations of all σ_i were fixed by the initial settings $s_i^{(0)}$ and only the common factor σ was adapted.

[3] As a notational convention, we drop the use of all projections of vectors to yield single components and simply define the notation

$$f_\sigma(\vec{x}, \sigma) = (\vec{x}, f_\sigma(\sigma)) \quad , \quad f_x(\vec{x}, \sigma) = (f_x(\vec{x}), \sigma) \quad ,$$

where

$$\tilde{\sigma} := \mathbf{mu}_\sigma(\sigma) = \begin{cases} \sigma / \sqrt[n]{c} & , \text{ if } p > 1/5 \\ \sigma \cdot \sqrt[n]{c} & , \text{ if } p < 1/5 \\ \sigma & , \text{ if } p = 1/5 \quad . \end{cases} \tag{6.6}$$

This definition reflects the 1/5-success rule after Rechenberg by updating the standard deviation σ at each generation[4], based on the measured relative frequency p of successful mutations. A choice of $c = 0.817$ was theoretically derived by Schwefel for the sphere model (see [Sch95], pp. 110–112).

The mutation of object variables proceeds by adding normally distributed variations with standard deviation $\tilde{\sigma}$ to the components of $\vec{x}$:

$$\tilde{x} := \mathbf{mu}_x(\vec{x}) = (x_1 + z_1, \ldots, x_n + z_n) \tag{6.7}$$

where[5] $z_i \sim \mathbf{N}_i(0, \tilde{\sigma}^2)$. Consequently, following (6.5) one obtains $\mathbf{mut}((\vec{x}, \sigma)) = \mathbf{mu}_x(\mathbf{mu}_\sigma(\vec{x}, \sigma)) = \mathbf{mu}_x(\vec{x}, \mathbf{mu}_\sigma(\sigma)) = \mathbf{mu}_x(\vec{x}, \tilde{\sigma}) = (\mathbf{mu}_x(\vec{x}), \tilde{\sigma}) = (\tilde{\vec{x}}, \tilde{\sigma})$.

The offspring individual $\tilde{\vec{a}} = (\tilde{\vec{x}}, \tilde{\sigma})$ and the parent $\vec{a} = (\vec{x}, \sigma)$ are both involved in the selection operator $\mathbf{sel}_1^2 : I^2 \to I$, which yields a surviving individual according to an objective function value comparison of $\vec{a}$ and $\tilde{\vec{a}}$:

$$\mathbf{sel}_1^2(\{\vec{a}, \tilde{\vec{a}}\}) = \begin{cases} \{\tilde{\vec{a}}\} & , \text{ if } f(\tilde{\vec{x}}) \le f(\vec{x}) \\ \{\vec{a}\} & , \text{ otherwise} \quad . \end{cases} \tag{6.8}$$

Using these mappings, the main loop of a (1+1)-ES reduces to

$$opt_{(1+1)-ES}(\{\vec{a}\}) = \mathbf{sel}_1^2(\{\mathbf{mut}(\vec{a})\} \sqcup \{\vec{a}\}) \quad , \tag{6.9}$$

and a pseudocode description of the algorithm is given below:

Algorithm 1 ((1 + 1)-ES)

```
$t := 0$;
initialize $P^{(t)} = \{(\vec{x}, \sigma)\}$;
evaluate $f(\vec{x})$;
while $(T(P^{(t)}) = 0)$ do            { $T$ denotes a termination criterion }
    $(\tilde{\vec{x}}, \tilde{\sigma}) := \mathbf{mut}((\vec{x}, \sigma))$;
    evaluate $f(\tilde{\vec{x}})$;        { determine objective function value }
    if $(f(\tilde{\vec{x}}) \le f(\vec{x}))$      { select }
        then $P^{(t+1)} := \{(\tilde{\vec{x}}, \tilde{\sigma})\}$;
        else $P^{(t+1)} := P^{(t)}$;
    $t := t + 1$;
od
```

and

$$(f_x \times g_\sigma)(\vec{x}, \sigma) = (f_x(\vec{x}), g_\sigma(\sigma)) \quad .$$

[4] An alternative consists in updating σ each n-th generation, using the constant c rather than $\sqrt[n]{c}$ as the update factor [Sch95].

[5] Assuming that different initial step sizes $s_i^{(0)}$ were defined, this has to be changed to $z_i \sim \mathbf{N}_i(0, (\tilde{\sigma} \cdot s_I^{(0)})^2)$.

Notice that the relative frequency p of successful mutations can easily be calculated by increasing a counter variable whenever selection chooses the offspring individual rather than the parent (i.e., a successful mutation has occurred). Schwefel formulates the corresponding algorithmic rule as follows (see [Sch95], p. 112):

> After every n mutations, check how many successes have occurred over the preceding $10 \cdot n$ mutations. If this number is less than $2 \cdot n$, multiply the step length by the factor $c = 0.85$; divide it by 0.85 if more than $2 \cdot n$ successes occurred.

The constant c was corrected by Schwefel in order to reflect the fact that the sphere model, for which the results were derived, is likely to require the fastest step size adaptation.

The (1+1)-ES is a kind of stochastic gradient technique, and the similarities to simulated annealing have been clarified by Rudolph [Rud93]. Though the algorithm is often useful, it is basically a local search strategy, and the 1/5 success rule may cause premature stagnation of the search due to the deterministic decrease of the step size whenever the topological situation does not lead to a sufficiently large success rate. One attempt to circumvent this problem, the (μ+1)-strategy, introduced the population concept on the parent level and therefore facilitated the development of the ($\mu \overset{+}{,} \lambda$)-strategies.

6.3 THE $(\mu + 1)$-EVOLUTION STRATEGY

Besides the (1+1)-ES, Rechenberg also proposed the first multimembered evolution strategy, a (μ+1)-ES (see [Rec73], chapter 9) with $\mu > 1$. The strategy was only outlined by Rechenberg, and he did not clarify how the modification of σ might work in this algorithm: Self-adaptation as used in the (μ,λ)-strategy definitely does not work in a (μ+1)-strategy, and it is also not clear how the 1/5-success rule might be applied in the (μ+1)-case, because the theoretical derivation only holds for the single parent strategy. Nevertheless, it is worthwhile to describe the algorithm here because it allows to introduce recombination in a straightforward way.

The recombination operator **rec** : $I^\mu \rightarrow I$ is applied before mutation to create a single individual from the parent population. The recombined individual undergoes mutation, and the resulting offspring individual substitutes the worst individual of the parent population if it performs at least as well as the worst parent (elimination of the worst). In other words, the μ best individuals out of μ parents and one offspring are deterministically selected as the next parent population. This kind of selection strategy which substitutes (at most) one individual per iteration of the evolutionary loop (elimination of the worst instead of survival of the fittest) is termed *steady-state selection* in genetic algorithm research [Whi89].

In contrast to genetic algorithms, the recombination operator **rec** : $I^\mu \rightarrow I$ creates just one individual per application. It works by first choosing ϱ ($1 \leq \varrho \leq \mu$) parent vectors from $P^{(t)} \in I^\mu$ with uniform probability, and then mixing characters from the ϱ parents to create one offspring vector:

$$\mathbf{rec} = \mathbf{re} \circ \mathbf{co} \quad , \tag{6.10}$$

where $\mathbf{co} : I^\mu \rightarrow I^\varrho$ chooses ϱ parent vectors and $\mathbf{re} : I^\varrho \rightarrow I$ creates one offspring vector. The cases $\varrho = 2$ of bisexual recombination as well as $\varrho = \mu$ of global recombination are commonly applied, but other settings of ϱ are equally possible[6].

Depending on the recombination type ω, a variety of ways exist to recombine the parental vectors in order to create an offspring. The recombination type may differ for the various parts of a complete individual $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha})$ of the $(\mu \overset{+}{,} \lambda)$-ES (see section 6.4 for the meaning of the components $\vec{\sigma}$ and $\vec{\alpha}$), such that we define the recombination types by referring to arbitrary vectors $\vec{b}$ and $\vec{b}'$, where $\vec{b}'$ denotes the part of an offspring vector to be generated and $b_{k,i}$ denotes the i-th component of a preselected individual number $k \in \{1, \ldots, \varrho\}$ out of the set of individuals chosen by $\mathbf{co}$. The original $(\mu$+1)-ES applies recombination to the object variable vector $\vec{x}$ only.

The most commonly used recombination types are:

- $\omega = 0$: No recombination (holds always when $\mu = 1$ or $\rho = 1$). In this case, only a random choice $\mathbf{co} : I^\mu \rightarrow I$ of a single individual is performed, and $\mathbf{re} = \mathbf{id} : I \rightarrow I$ is just the identity mapping.
- $\omega = 1$: Global intermediary recombination, where the i-th vector component is averaged over all parents to obtain the corresponding offspring component value:

$$b_i' = \frac{1}{\varrho} \sum_{k=1}^{\varrho} b_{k,i} \tag{6.11}$$

- $\omega = 2$: Local intermediary recombination, which works by selecting two out of the ϱ parents for each vector component and calculating a weighted sum of the corresponding components of the two parents:

$$b_i' = u_i b_{k_1,i} + (1 - u_i) b_{k_2,i} \quad , \tag{6.12}$$

  where $u_i \sim \mathbf{U}([0,1])$ or $u_i = 1/2$, and $k_1, k_2 \sim \mathbf{U}(\{1, \ldots, \varrho\})$ for each offspring.
- $\omega = 3$: Discrete recombination, where each vector component is copied from the corresponding component of an individual randomly chosen from the ϱ parents:

$$b_i' = b_{k_i,i} \tag{6.13}$$

  where $k_i \sim \mathbf{U}(\{1, \ldots, \varrho\})$ at random for each i.

Of course, other recombination operators such as multiple point crossover as commonly used in genetic algorithms are also possible and could be added to the set of available operators.

Following equation (6.4), the main loop of the $(\mu$+1)-ES is formulated as follows:

$$opt_{(\mu+1)-ES}(P^{(t)}) = \mathbf{sel}_\mu^{\mu+1}(\{\mathbf{mut}(\mathbf{rec}(P^{(t)}))\} \sqcup P^{(t)}) \quad . \tag{6.14}$$

The selection operator simply yields the set of the μ best individuals of its argument, i.e., $\mathbf{sel}_\mu^k(P) = \tilde{P}$, where $|\tilde{P}| = \mu$, $|P| = k \geq \mu$, and

$$\forall \tilde{\vec{a}} \in \tilde{P} : \not\exists \vec{a} \in P - \tilde{P} : \quad f(\vec{x}) \leq f(\tilde{\vec{x}}) \quad . \tag{6.15}$$

[6] Recently, this concept has also been tested successfully in the context of genetic algorithm research [ERR95].

Algorithm 2 presents a pseudocode description of the (μ+1)-ES as defined here formally.

Algorithm 2 (($\mu + 1$)-ES)

```
    $t := 0;$
    initialize $P^{(0)} = \{\vec{x}_1, \ldots, \vec{x}_\mu\} \in I^\mu;$
    evaluate $f(\vec{x}_1), \ldots, f(\vec{x}_\mu);$
    while $(T(P^{(t)}) = 0)$ do
        $\tilde{\vec{x}} := \mathbf{mut}(\mathbf{rec}(P^{(t)}));$
        evaluate $f(\tilde{\vec{x}});$
        $P^{(t+1)} := \mathbf{sel}_\mu^{\mu+1}(\{\tilde{\vec{x}}\} \sqcup P^{(t)});$
        $t := t + 1;$
    od
```

The implementation of recombination is usually based on switching according to the recombination type ω. The parental individuals are chosen by means of a uniform random number generator.

As indicated before, the (μ+1)-strategy does not offer an appropriate method to control the standard deviation σ, such that we did not further specify the structure of individuals and the working mechanism of the mutation operator. Even if the 1/5 success rule could be adapted to work, the strategy would still suffer from the disadvantages of this rule. The modern (μ,λ)-strategy, which is discussed in the next section, solves this problem by self-adapting the standard deviation(s) on-line during the search.

6.4 THE (μ, λ)- AND $(\mu + \lambda)$-EVOLUTION STRATEGY

The severe disadvantages of the 1/5 success rule for controlling the "step size" σ of the simple evolution strategy caused Schwefel to look for a more robust and general method to adjust the mutation parameters of the algorithm. Again, a solution to this problem was found by taking a closer look at the natural model, where the genotype itself incorporates mechanisms to control its own mutability (by means of genotype segments that encode repair enzymes, or by so-called mutator genes). Transferring this to the evolution strategy means that the standard deviation for mutation becomes part of the individual and evolves by means of mutation and recombination just as the object variables do — a process called self-adaptation of strategy parameters[7] [Sch77].

More precisely, individuals in a (μ,λ)-strategy are equipped with a set of strategy parameters which represent an n-dimensional normal distribution for mutating the individual:

$$I = \boldsymbol{R}^n \times \boldsymbol{R}_+^{n_\sigma} \times [-\pi, \pi]^{n_\alpha} \quad , \tag{6.16}$$

i.e., $S = \boldsymbol{R}^{n_\sigma} \times [-\pi, \pi]^{n_\alpha}$.

An individual $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha}) \in I$ consists of the components

[7] Sometimes, the terms auto-adaptation or second-level learning are also used to denote this principle of evolving strategy parameters.

- $\vec{x} \in \boldsymbol{R}^n$: The vector of object variables. Notice that this is the only part of $\vec{a}$ entering the objective function.
- $\vec{\sigma} \in \boldsymbol{R}_+^{n_\sigma}$: A vector of standard deviations ($1 \leq n_\sigma \leq n$) of the normal distribution.
- $\vec{\alpha} \in [-\pi, \pi]^{n_\alpha}$: A vector of inclination angles ($n_\alpha = (n - n_\sigma/2) \cdot (n_\sigma - 1)$), defining linearly correlated mutations of the object variables $\vec{x}$.

The strategy parameters $\vec{\sigma}$ and $\vec{\alpha}$ determine the variances and covariances of the n-dimensional normal distribution, which is used for exploring the search space (see part II for a discussion of the n-dimensional normal distribution and its properties).

The amount of strategy parameters attached to an individual can be varied by the user of an evolution strategy, depending on her feelings about the degree of freedom required by the objective function topology. As a rule of thumb, the global search reliability and robustness of the algorithm increases at the cost of computing time when the number of strategy parameters is increased. The settings most commonly used are:

- $n_\sigma = 1$, $n_\alpha = 0$: Standard mutations with one single standard deviation controlling mutation of all components of $\vec{x}$.
- $n_\sigma = n$, $n_\alpha = 0$: Standard mutations with individual step sizes $\sigma_1, \ldots, \sigma_n$ controlling mutation of the corresponding object variables x_i individually.
- $n_\sigma = n$, $n_\alpha = n \cdot (n-1)/2$: Correlated mutations with a complete covariance matrix for each individual. Note that in this case the correspondence $\Delta x_i \propto \sigma_i$ is no longer valid.
- $n_\sigma = 2$, $n_\alpha = n - 1$: In one arbitrary direction of the search space the search is performed with variance σ_1^2 whereas σ_2^2 is the variance in all other directions perpendicular to the first one.

The basic idea of correlated mutations is illustrated for the case $n = 2$, $n_\sigma = 2$, $n_\alpha = 1$ in figure 6.1, where the lines of equal mutation probability density of the two-dimensional normal distribution are plotted. Notice that the standard deviations σ_1 and σ_2 determine the relation of the lengths of the main axes of the hyperellipsoid, and α_{12} represents the rotation angle of the hyperellipsoid. In the general case of correlated mutations, the mutation hyperellipsoid may align itself arbitrarily in the n-dimensional search space.

According to the generalized structure of individuals, the mutation operator $\mathbf{mut} : I \rightarrow I$ is defined as follows[8]:

$$\mathbf{mut} = \mathbf{mu}_x \circ (\mathbf{mu}_\sigma \times \mathbf{mu}_\alpha) \quad . \tag{6.17}$$

The operator is applied after recombination to an individual

$$\hat{\vec{a}} = (\hat{x}_1, \ldots, \hat{x}_n, \hat{\sigma}_1, \ldots, \hat{\sigma}_{n_\sigma}, \hat{\alpha}_1, \ldots, \hat{\alpha}_{n_\alpha})$$

and proceeds by first mutating the strategy parameters $\hat{\vec{\sigma}}$ and $\hat{\vec{\alpha}}$ and then modifying $\vec{x}$ according to the new set of strategy parameters obtained from mutating $\hat{\vec{\sigma}}$ and $\hat{\vec{\alpha}}$:

[8] Again, as a notational convention, we define

$$(\mathbf{mu}_\sigma \times \mathbf{mu}_\alpha)(\vec{x}, \vec{\sigma}, \vec{\alpha}) = (\vec{x}, \mathbf{mu}_\sigma(\vec{\sigma}), \mathbf{mu}_\alpha(\vec{\alpha})) \quad .$$

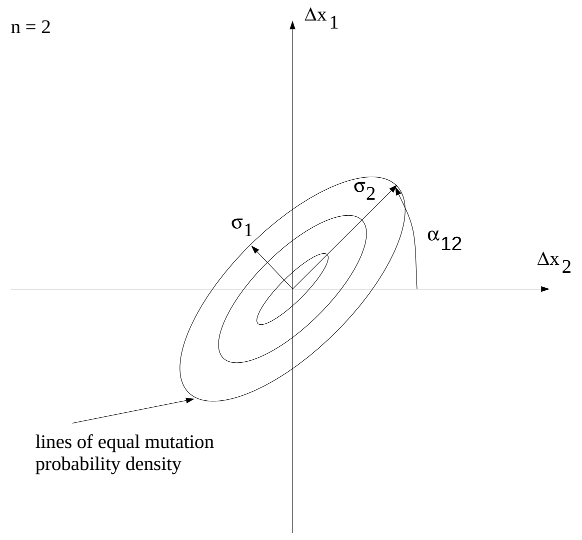

Figure 6.1: Illustration of the mutation ellipsoid for the case $n = 2$ (number of object variables), $n_\sigma = 2$ (number of variances), $n_\alpha = 1$ (number of covariances).

- $\mathbf{mu}_\sigma : \boldsymbol{R}_+^{n_\sigma} \to \boldsymbol{R}_+^{n_\sigma}$ mutates the recombined $\hat{\vec{\sigma}}$:

$$\mathbf{mu}_\sigma(\hat{\vec{\sigma}}) := (\hat{\sigma}_1 \exp(z_1 + z_0), \ldots, \hat{\sigma}_{n_\sigma} \exp(z_{n_\sigma} + z_0)) =: \tilde{\vec{\sigma}} \qquad (6.18)$$

  where $z_0 \sim \mathbf{N}(0, \tau_0^2)$, $z_i \sim \mathbf{N}(0, \tau^2)\ \forall i \in \{1, \ldots, n_\sigma\}$.
  To prevent the standard deviations from becoming practically zero, a minimal value of ε_σ is algorithmically enforced for all σ_i.
- $\mathbf{mu}_\alpha : \boldsymbol{R}^{n_\alpha} \to \boldsymbol{R}^{n_\alpha}$ mutates the recombined $\hat{\vec{\alpha}}$:

$$\mathbf{mu}_\alpha(\hat{\vec{\alpha}}) := (\hat{\alpha}_1 + z_1, \ldots, \hat{\alpha}_{n_\alpha} + z_{n_\alpha}) =: \tilde{\vec{\alpha}} \qquad (6.19)$$

  where $z_i \sim \mathbf{N}(0, \beta^2)\ \forall i \in \{1, \ldots, n_\alpha\}$. Empirically, $\beta \approx 0.0873$ ($\approx 5°$) has shown to yield good results.
  Rotation angles are kept feasible (i.e., in the interval $[-\pi, \pi]$) by circularly mapping them into the feasible range whenever it is left by mutation.
- $\mathbf{mu}_x : \boldsymbol{R}^n \to \boldsymbol{R}^n$ mutates the recombined object variable vector $\vec{x}$, using the already mutated $\tilde{\vec{\sigma}}$ and $\tilde{\vec{\alpha}}$:

$$\mathbf{mu}_x(\vec{x}) := (\hat{x}_1 + cor_1(\tilde{\vec{\sigma}}, \tilde{\vec{\alpha}}), \ldots, \hat{x}_n + cor_n(\tilde{\vec{\sigma}}, \tilde{\vec{\alpha}})) =: \tilde{\vec{x}} \qquad (6.20)$$

  where $\vec{cor} := (cor_1(\tilde{\vec{\sigma}}, \tilde{\vec{\alpha}}), \ldots, cor_n(\tilde{\vec{\sigma}}, \tilde{\vec{\alpha}}))$ is a random vector with normally distributed, eventually correlated components. The vector $\vec{cor}$ can be calculated according to $\vec{cor} = \mathbf{T}\vec{z}$ where $\vec{z} = (z_1, \ldots, z_n)$ with

$z_i \sim \mathbf{N}(0, \tilde{\sigma}_i^2) \; \forall i \in \{1, \ldots, n_\sigma\}$ and

$$\mathbf{T} = \prod_{p=1}^{n_\sigma - 1} \prod_{q=p+1}^{n_\sigma} \mathbf{T}_{pq}(\tilde{\alpha}_j) \tag{6.21}$$

with $j = \frac{1}{2}(2n_\sigma - p)(p+1) - 2n_\sigma + q$ [Rud92a]. The rotation matrices $\mathbf{T}_{pq}(\tilde{\alpha}_j)$ are unit matrices except that $t_{pp} = t_{qq} = \cos(\alpha_j)$ and $t_{pq} = -t_{qp} = -\sin(\alpha_j)$, i.e., the trigonometric terms are located in columns p and q, each. The multiplication from right to left provides an efficient way of calculating (6.21).

The use of a logarithmic normal distribution for the variation of standard deviations σ_i is motivated by the arguments that a multiplicative process guarantees standard deviations to remain positive, that the median equals one (i.e., the process is neutral when selection is disabled), and that smaller modifications are more likely than larger ones (see [Sch77], p. 168).

For recombination, all possibilities as described in section 6.3 for $\omega \in \{0,1,2,3\}$ can be used in principle. The recombination operator modifies not only the object variables, but also the strategy parameters, and the operator might be different for the components $\vec{x}$, $\vec{\sigma}$, and $\vec{\alpha}$ of an individual. Consequently, the complete recombination operator is specified by vectors $\vec{\omega} = (\omega_x, \omega_\sigma, \omega_\alpha) \in \{0,1,2,3\}^3$ and $\vec{\varrho} = (\varrho_x, \varrho_\sigma, \varrho_\alpha) \in \{1, \ldots, \mu\}^3$, where ω_x, ω_σ, and ω_α specify the recombination operator for object variables, standard deviations, and rotation angles, and ϱ_x, ϱ_σ, and ϱ_α are the number of potential parents involved in the recombination of object variables, standard deviations, and rotation angles. Consequently, recombination splits into three separate operator combinations for $\vec{x}$, $\vec{\sigma}$, and $\vec{\alpha}$, following equation (6.10):

$$\mathbf{rec} = (\mathbf{re}_x \circ \mathbf{co}_x) \times (\mathbf{re}_\sigma \circ \mathbf{co}_\sigma) \times (\mathbf{re}_\alpha \circ \mathbf{co}_\alpha) \quad , \tag{6.22}$$

where $\mathbf{co}_\delta : I^\mu \to I^{\varrho_\delta}$ and $\mathbf{re}_\delta : I^{\varrho_\delta} \to I$ $(\delta \in \{x, \sigma, \alpha\})$.

Notice that, according to the separate application of potentially different recombination operators on the components of individuals, an object variable x_i and a standard deviation σ_i are usually not transferred to the offspring together, as a unit of information. Sometimes, the choice of a useful recombination operator for a particular optimization problem is relatively difficult and requires to perform some experiments, but in many cases the setting[9] $\vec{\omega} = (3,2,0)$ with $\vec{\varrho} = (\mu, \mu, 1)$ provides an appropriate parameterization.

Combining the operators mutation, recombination, and selection as defined here, the main loop of a (μ,λ)-ES is formulated according to equation (6.4) as follows:

$$opt_{(\mu,\lambda)-ES}(P^{(t)}) = \mathbf{sel}_\mu^\lambda(\sqcup_{i=1}^{\lambda}\{\mathbf{mut}(\mathbf{rec}(P^{(t)}))\}) \quad . \tag{6.23}$$

According to definition (6.15), selection simply returns the set consisting of the μ best individuals of its argument set of size λ.

[9] I.e., discrete recombination on two randomly selected parents on $\vec{x}$ and local intermediary recombination using the whole parent population as a gene pool for the variances and no recombination for the inclination angles. Typically, the setting $u_i = 1/2$ is used for local intermediary recombination.

In case of a (μ+λ)-ES, the main loop changes only slightly by taking into account the parent population, too, i.e., the argument set of **sel** has size $\lambda + \mu$:

$$opt_{(\mu+\lambda)-ES}(P^{(t)}) = \mathbf{sel}_{\mu}^{\mu+\lambda}(\sqcup_{i=1}^{\lambda}\{\mathbf{mut}(\mathbf{rec}(P^{(t)}))\} \sqcup P^{(t)}) \quad . \tag{6.24}$$

Though it offers some theoretical advantage (see part II), this minor modification has the serious disadvantage that the self-adaptation of strategy parameters is hindered in working effectively, because misadapted strategy parameters may survive for a relatively large number of generations. Furthermore, the (μ+λ)-selection mechanism fails in case of dynamically changing environments, and it tends to emphasize on local rather than global search properties. For these reasons, modern evolution strategies use (μ,λ)-selection, normally. Algorithm 3 presents the (μ,λ)-ES in pseudocode notation:

Algorithm 3 ((μ, λ)-ES)

```
$t := 0$;
initialize $P^{(0)} = \{\vec{a}_1, \ldots, \vec{a}_\mu\} \in I^\mu$;
evaluate $f(\vec{x}_1), \ldots, f(\vec{x}_\mu)$;
while ($T(P^{(t)}) = 0$) do
    $\tilde{P} := \emptyset$;
    for $i := 1$ to $\lambda$ do
        $(\tilde{\vec{x}}, \tilde{\vec{\sigma}}, \tilde{\vec{\alpha}}) := \mathbf{mut}(\mathbf{rec}(P^{(t)}))$;
        evaluate $f(\tilde{\vec{x}})$;
        $\tilde{P} := \tilde{P} \sqcup \{(\tilde{\vec{x}}, \tilde{\vec{\sigma}}, \tilde{\vec{\alpha}})\}$;
    od
    $P^{(t+1)} := \mathbf{sel}_{\mu}^{\lambda}(\tilde{P})$;
    $t := t + 1$;
od
```

The self-adaptation process of strategy parameters is based on the existence of an indirect link between strategy parameters (the internal model of the individual) and the fitness of an individual as well as a sufficiently large diversity of internal models in the parent population: μ has to be chosen clearly larger than one, e.g. $\mu = 15$ [Sch92], and a ratio of $\lambda/\mu \approx 7$ is recommended as a good setting for the relation between parent and offspring population size[10]. So far, no theoretical results about the self-adaptation of strategy parameters have been obtained, but empirically three main conditions for a successful self-adaptation were identified [Sch92]:

- (μ,λ)-selection,
- a not too strong selective pressure (i.e., μ has to be clearly larger than one),
- recombination on strategy parameters.

To conclude this description of the (μ,λ)-ES, we have to mention the topics of initialization, constraint handling and termination criteria. So far, the basic method to handle constraints consists in repeating the processes of recombination and mutation as often as necessary to create λ feasible offspring individuals, i.e., with $g_j(\tilde{\vec{x}}) \geq$

[10] Typically, a (15,100)-ES is used.

$0 \ \forall j \in \{1, \ldots, q\}$. It is often necessary, however, to use a more intelligent method such as penalty terms, for instance, and constraint handling is a general topic of strong importance for all instances of evolutionary algorithms.

The simplest termination criterion $T : I^{\mu} \rightarrow \{0, 1\}$ stops evolution after a previously specified number of generations has passed, independently of the population diversity or the number of generations required for the last improvement that occurred. Alternative methods as proposed by Schwefel terminate the search if the objective function value improvement over a number of generations or the fitness difference between worst and best individuals falls below a certain threshold (see [Sch95], pp. 113-114).

Concerning initialization of the start population $P^{(0)}$, two possibilities are useful for comparing evolution strategies to classical optimization methods as well as genetic algorithms. First, with given lower and upper bounds $\underline{x}_i, \overline{x}_i \in \boldsymbol{R}$, i.e., $\underline{x}_i \leq x_i \leq \overline{x}_i \ \forall i \in \{1, \ldots, n\}$, all individuals of $P^{(0)}$ are arbitrarily (most often uniformly) distributed within the bounded region. Second, one might also know a given start position $\vec{x}$, which is then assigned to one individual, and the remaining $\mu - 1$ individuals are generated by mutation of this individual with some enlarged step size $c \cdot \sigma^{(0)}$, $c > 1$.

The description presented in this section is certainly sufficient for the reader to implement a (μ,λ)-evolution strategy. More details on the realization of specific operators, however, can be found in [Sch95].

6.5 FURTHER VARIANTS OF THE EVOLUTION STRATEGY

The concept of self-adaptation of strategy parameters by means of extending the evolutionary operators mutation and recombination to the step sizes $\vec{\sigma}$ (and, later on, the covariances represented by rotation angles $\vec{\alpha}$) was introduced by Schwefel, who demonstrated the robustness and learning capabilities of the mechanism [Sch87, Sch92]. For the restricted case $n_{\sigma} = 1$, Rechenberg presented a simple alternative method called mutative step-size control, which proceeds by multiplying σ by a factor c' or $1/c'$ with equal probability ([Rec94], pp. 45-50). For an individual $\vec{a} = (\vec{x}, \sigma)$, the mutation operator $\mathbf{mu}_{\sigma}$ works as follows:

$$\tilde{\sigma} := \mathbf{mu}_{\sigma}(\sigma) = \begin{cases} \sigma \cdot c' & , \ \text{if } u < 1/2 \\ \sigma / c' & , \ \text{otherwise} \end{cases} \tag{6.25}$$

where $u \sim \mathbf{U}([0, 1])$. For the constant c', Rechenberg suggests a value of 1.3. Mutation of the object variables is performed as in equation (6.7). Rechenberg confirms Schwefel's observation that a $(\mu+\lambda)$-selection hampers the self-adaptation of strategy parameters, which is also true for the mutative step-size control. Although easy to implement, the mutative step-size control is insufficient for complicated optimization problems, because the internal models of individuals are too restricted by $n_{\sigma} = 1$ to explore arbitrary n-dimensional search spaces in an effective way.

Adapting a relatively large number of strategy parameters, however, requires to work with sufficiently large populations, which is hard to realize if the evaluation of objective function values is a computationally expensive task. Concerning correlated mutations, Rudolph already pointed out that it is worthwhile to make use of the history of the evolution process, e.g. by storing the phylogenetic tree of individuals up to a depth of n [Rud92a]. An accumulation of step-size information over the sequence of generations is also used by Ostermeier et al. to introduce a derandomized mutative step-size control of individual step sizes $\sigma_1, \ldots, \sigma_n$ in a (1,λ)-ES [OGH94]. These are interesting developments towards improving and generalizing the self-adaptation process, but more experience as well as comparisons of the different implementations and theoretical investigations are necessary to assess their pros and cons. And, moreover, the question of a proper test suite arises and whether one emphasizes robustness or efficiency.

Besides modeling evolution on the level of a single population, it is also possible to model the evolution of a species by introducing a higher-level competition between populations. Herdy uses the notation $[\mu'/\varrho', \lambda'(\mu/\varrho, \lambda)^\gamma]$ to describe an evolution scheme where μ' parent populations generate λ' offspring populations (involving ϱ' parent populations for creating one offspring population), and each population follows a (μ,λ)-evolution strategy[11] [Her92]. A recombination-mutation-selection step on the level of populations happens after each γ generations that have passed by on the level of individuals. Rechenberg points out that complete populations might be evaluated according to their average objective function value, but other measures of quality are also possible (see [Rec94], pp. 88-100). Both authors also indicate that even higher levels of hierarchical evolution strategies are possible.

The major advantage of evolving several populations in parallel and exchanging information between these populations only occasionally is clearly given by the higher convergence reliability in case of highly multimodal optimization problems: Such parallel algorithms with k (μ,λ)-evolution strategies usually succeed in finding better solutions than a single $(k \cdot \mu, k \cdot \lambda)$-strategy, because the subpopulations explore different regions of the search space and maintain genetic diversity by communicating information not too often to other subpopulations. In general, four additional parameters control a collection of parallel interacting subpopulations (also called demes): Exchange frequency, number of individuals to exchange, selection strategy for the emigrants, and replacement strategy to incorporate the immigrants. Due to the exchange of individuals, this is also often called a migration model. Such a migration-based evolution strategy (or evolutionary algorithm in general) is suitable for implementation on a coarse-grained parallel computer with a low communication bandwidth relative to the computational performance of each processor. An evolution strategy with asynchronously communicating parallel populations was presented by Rudolph for an application to a 100-city TSP [Rud91].

Besides parallel populations, the inherent parallelism of evolutionary algorithms can also be exploited on the level of individuals. The corresponding fine-grained parallel algorithm uses a neighbourhood structure (often a toroidal grid), where individuals interact only with other individuals in their neighborhood, i.e., recombination and selection are restricted to the neighborhood. In this model, favourable genetic information might spread over the population due to overlapping neighborhoods, and for this reason the model is sometimes called diffusion model. The corresponding implementation for an evolution strategy was also performed by Rudolph on a Connection Machine 2 [Rud92b].

Fine-grained parallel computers typically allow for a large number of processes of

[11] Additionally, μ and λ may differ for each of the λ' subpopulations.

low complexity, each, and have to provide a high communication bandwidth. Such machines are not suitable for implementing the migration model, where each process consists of a complete evolutionary algorithm. On the contrary, coarse-grained parallel machines with a small number of powerful processors allow to emulate a fine-grained neighborhood structure and to combine the migration and diffusion model in an elegant way, as shown by Sprave [Spr90, Spr93].

Finally, it is important to mention that the most time-consuming component of an evolution strategy may consist in the evaluation of objective function values or of the constraints — which is often the case for practical applications, especially when evaluating f and g_j requires to run a complete simulation model [BHS93]. Under such circumstances, it is an elegant solution to evaluate each function value on a single workstation in a local area network, because the communication requirements are small (only $\vec{x}$ and $f(\vec{x})$ have to be exchanged) and the requirement concerning computing power (and potentially other hardware resources such as main memory or disk storage) is high. Presently, the PVM software provides an elegant, hardware-independent method to create a virtual parallel computer by using a (heterogeneous) network of machines [GBD$^+$94]. For a more detailed overview of the parallelization possibilities of evolutionary algorithms, the reader is referred to the work of Hoffmeister [Hof91].

Modifications of the original (μ,λ)-ES have also been developed for applying the algorithm to multiple criteria decision making (MCDM) problems (Kursawe presented a variant which is capable of generating arbitrarily many members of the Pareto-set by varying the selection criterion and using diploid individuals [Kur91]) and mixed-integer optimization (Bäck and Schütz discuss the extension of self-adaptation to discrete variables and the application of the algorithm to the problem of optical multilayer design [BS95]). The reader is referred to the literature for a more detailed presentation of these variants.

6.6 OUTLOOK

Over a period of thirty years, evolution strategies have evolved and are now in a state where a variety of different variants are known and research continues into several directions, including applications, extensions of algorithms, theory, and cross-fertilization with other evolutionary algorithms. The theory of (μ,λ)-evolution strategies has reached a relatively mature state — although a lot of particular questions are still open (see the contribution "Evolution Strategies II: Theoretical Aspects" in this volume).

Concerning the algorithm itself, a contemporary evolution strategy called $(\mu, \kappa, \lambda, \varrho)$-ES was recently proposed by Schwefel and Rudolph [SR95]. The algorithm extends the (μ,λ)-ES by some new features, including:

- The life span of individuals is limited to $\kappa \geq 1$ generations (better: reproduction cycles), which allows a free variation of the selection scheme between the extreme cases $\kappa = 1$ ((μ,λ)-selection) and $\kappa = \infty$ ($(\mu+\lambda)$-selection). This is implemented by extending the individuals by a life span counter θ, which is initialized by $\theta = \kappa$ at $t = 0$ and whenever an offspring individual is born by recombination and mutation. The selection operator chooses an individual to survive only, if its remaining life span θ is larger than zero — otherwise, an individual which is worse with respect to fitness will survive. The selection condition (6.15) therefore changes to

$$\forall \tilde{\vec{a}} \in \tilde{P} : \theta_{\tilde{\vec{a}}} > 0 \wedge (\nexists \vec{a} \in P - \tilde{P} : \theta_{\vec{a}} > 0 \wedge f(\vec{x}) \leq f(\tilde{\vec{x}})) \quad . \qquad (6.26)$$

At the end of the selection process, the remaining life durations are decremented by one for each of the μ surviving individuals.

- Tournament selection is incorporated as an alternative to (μ,λ)-selection. This method is well suited for parallelization and works by selecting μ times the best individual from a subset B_k of size $|B_k| = \zeta$ (tournament size, $2 \leq \zeta \leq \mu + \lambda$), $k \in \{1, \ldots, \mu\}$, whose elements are chosen uniformly at random from $P = \sqcup_{i=1}^{\lambda} \{\mathbf{mut}(\mathbf{rec}(P^{(t)}))\} \sqcup P^{(t)}$ (note that duplicates are explicitly allowed to occur in the B_k). The union of the best members of these μ subsets forms the new parent population.
- The application of recombination and mutation is controlled by additional probabilities $p_m, p_r \in [0, 1]$, such that the operators are applied with probability p_m and p_r, respectively.
- Further recombination types such as crossover from genetic algorithms are incorporated.

The ($\mu, \kappa, \lambda, \varrho$)-ES reflects the attempt to incorporate useful features from genetic algorithms and to benefit from modeling the natural concept of limited life spans of individuals in a more flexible way than in a (μ,λ)-ES with their one-generation restriction of life spans. The algorithm provides a generalization of the state-of-the-art evolution strategy, and certainly many experiments are necessary to investigate the effects of the additional features, especially concerning their impact on the self-adaptation of strategy parameters.

References

[BHS93] Bäck T., Hammel U., and Schwefel H.-P. (1993) Modelloptimierung mit evolutionären Algorithmen. In Sydow A. (ed) *Simulationstechnik: 8. Symposium in Berlin*, Fortschritte in der Simulationstechnik, pages 49–57. Vieweg, Wiesbaden.

[BS95] Bäck T. and Schütz M. (1995) Evolution strategies for mixed-integer optimization of optical multilayer systems. In *Proceedings of the 4th Annual Conference on Evolutionary Programming.* (accepted for publication).

[DSM94] Davidor Y., Schwefel H.-P., and Männer R. (eds) (1994) *Parallel Problem Solving from Nature — PPSN III, International Conference on Evolutionary Computation*, volume 866 of *Lecture Notes in Computer Science.* Springer, Berlin.

[ERR95] Eiben A. E., Raué P.-E., and Ruttkay Z. (1995) Genetic algorithms with multi-parent recombination. In Davidor *et al.* [DSM94], pages 78–87.

[GBD+94] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., and Sunderam V. (1994) *PVM: Parallel Virtual Machine — A User's Guide and Tutorial for Networked Parallel Computing.* The MIT Press, Cambridge, MA.

[HB92] Hoffmeister F. and Bäck T. (1992) Genetic self–learning. In Varela F. J. and Bourgine P. (eds) *Proceedings of the 1st European Conference on Artificial Life*, pages 227–235. The MIT Press, Cambridge, MA.

[Her92] Herdy M. (1992) Reproductive isolation as strategy parameter in hierarchically organized evolution strategies. In Männer and Manderick [MM92], pages 207–217.

[Hof91] Hoffmeister F. (1991) Scalable parallelism by evolutionary algorithms. In Grauer M. and Pressmar D. B. (eds) *Parallel Computing and Mathematical Optimization*, volume 367 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–198. Springer, Berlin.

[KS70] Klockgether J. and Schwefel H.-P. (March 24–26, 1970) Two–phase nozzle and hollow core jet experiments. In Elliott D. (ed) *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148. California Institute of Technology, Pasadena CA.

[Kur91] Kursawe F. (1991) A variant of Evolution Strategies for vector optimization. In Schwefel and Männer [SM91], pages 193–197.

[MM92] Männer R. and Manderick B. (eds) (1992) *Parallel Problem Solving from Nature 2.* Elsevier, Amsterdam.

[OGH94] Ostermeier A., Gawelczyk A., and Hansen N. (1994) Step-size adaptation based on non-local use of selection information. In Davidor *et al.* [DSM94], pages 189–198.

[Rec73] Rechenberg I. (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog, Stuttgart.

[Rec94] Rechenberg I. (1994) *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik.* frommann–holzboog, Stuttgart.

[Rud91] Rudolph G. (1991) Global optimization by means of distributed evolution strategies. In Schwefel and Männer [SM91], pages 209–213.

[Rud92a] Rudolph G. (1992) On correlated mutations in evolution strategies. In Männer and Manderick [MM92], pages 105–114.

[Rud92b] Rudolph G. (1992) Parallel approaches to stochastic global optimization. In Joosen W. and Milgrom E. (eds) *Parallel Computing: From Theory to Sound Practice, Proceedings of the European Workshop on Parallel Computing*, pages 256–267. IOS Press, Amsterdam.

[Rud93] Rudolph G. (1993) Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation* 1(4): 361–382.

[Sch77] Schwefel H.-P. (1977) *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, Basel.

[Sch87] Schwefel H.-P. (June 1987) 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.

[Sch92] Schwefel H.-P. (1992) Imitating evolution: Collective, two-level learning processes. In Witt U. (ed) *Explaining Process and Change — Approaches to Evolutionary Economics*, pages 49–63. The University of Michigan Press, Ann Arbor, MI.

[Sch95] Schwefel H.-P. (1995) *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. Wiley, New York.

[SM91] Schwefel H.-P. and Männer R. (eds) (1991) *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, volume 496 of *Lecture Notes in Computer Science*. Springer, Berlin.

[Spr90] Sprave J. (December 1990) *Parallelisierung Genetischer Algorithmen zur Suche und Optimierung*. Diploma thesis, University of Dortmund.

[Spr93] Sprave J. (1993) Zelluläre evolutionäre Algorithmen zur Parameteroptimierung. In Hofestädt R., Krückeberg F., and Lengauer T. (eds) *Informatik in den Biowissenschaften*, Informatik aktuell, pages 111–120. Springer, Berlin.

[SR95] Schwefel H.-P. and Rudolph G. (1995) Contemporary evolution strategies. In Morán F., Moreno A., Merelo J. J., and Chacón P. (eds) *Advances in Artificial Life. Third International Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 893–907. Springer, Berlin.

[Whi89] Whitley D. (1989) The GENITOR algorithm and selection pressure: Why rank–based allocation of reproductive trials is best. In Schaffer J. D. (ed) *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 116–121. Morgan Kaufmann Publishers, San Mateo, CA.