Chapter 2
Evolution Strategies

2.1 Introduction

Many real-world problems are multimodal, which renders an optimization problem difficult to solve. Local search methods, i.e., methods that greedily improve solutions based on search in the neighborhood of a solution, often only find an arbitrary local optimum that is not guaranteed to be the global one. The most successful methods in global optimization are based on stochastic components, as they allow to escape from local optima and overcome premature stagnation. A famous class of global optimization methods are evolution strategies that are successful in real-valued solution spaces. Evolution strategies belong to the most famous evolutionary methods for black box optimization, i.e., for optimization scenarios, where no functional expressions are explicitly given and no derivatives can be computed. In the course of this work, evolution strategies will play an important role. They are oriented to the biological principle of evolution [1] and can serve as an excellent starting point to methods in learning and optimization. They are based on three main mechanisms that are translated into evolutionary operators:

1. recombination,
2. mutation, and
3. selection.

First, we define an optimization problem formally. Let $f : S \rightarrow \mathbb{R}$ be the fitness function to be minimized in the space of solutions S. The problems we consider in this work are minimization problems unless explicitly stated. *High* fitness means *low* fitness values. The task is to find an element $\mathbf{x}^* \in S$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$. A desirable property of an optimization method is to find the optimum $\mathbf{x}^*$ with fitness f^* within a finite and preferably low number of function evaluations. In most parts of this work, we consider continuous optimization problems, i.e., the solution space $S = \mathbb{R}^N$. Problem f can be an arbitrary optimization problem, e.g., a civil engineering system like a simulation or a mathematical model.

DOI: 10.1007/978-3-319-03422-5_2, © The Author(s) 2014

2.2 Evolutionary Algorithms

If derivatives are available, Newton methods and variants are recommendable algorithmic choices. From this class of methods, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm belongs to the state-of-the-art techniques [2]. This work concentrates on black box optimization problems. In black box optimization, the problem does not have to fulfill any assumptions or limiting properties. For such general optimization scenarios, evolutionary methods are a good choice. Evolutionary algorithms (EAs) belong to the class of stochastic derivative-free optimization methods. Their biological motivation has made them very popular. After decades of research, a long history of applications and theoretical investigations have proven their success.

In Germany, the history of evolutionary computation began with evolution strategies, which were developed by Rechenberg and Schwefel in the sixties and seventies of the last century in Berlin [3–5]. At the same time, John Holland introduced the evolutionary computation concept in the United States known as *genetic algorithms* [6]. Today, advanced mutation operators, step size mechanisms, and methods to adapt the covariance matrix like the CMA-ES [7] have made them one of the most successful optimizers in derivative-free global optimization.

Many methods have been presented in evolutionary continuous optimization like the work by Deb et al. [8], who developed a generic parent-centric crossover operator, and a steady-state, elite-preserving population-alteration model. Herrera et al. [9, 10] proposed to apply a two-loop EA with adaptive control of mutation sizes. The algorithm adjusts the step size of an inner EA and a restart control of a mutation operator in the outer loop. Differential evolution (DE) is another branch of evolutionary methods for continuous optimization. Price et al. [11] give an introductory survey to DE. Qin et al. [12] proposed an adaptive DE that learns operator selection and associated control parameter values. The learning process is based on previously generated successful solutions. Particle swarm optimization (PSO) is a famous methodology that concentrates on continuous global optimization [13, 14]. PSO is inspired by the movement of swarms in nature, e.g., fish schools or flocks of birds. It simulates the movement of candidate solutions using flocking-like equations with locations and velocities. A learning strategy variant has been proposed by Liang et al. [15], who uses all particles' past best information to update the particle history. A PSO-like algorithm will be employed in Chap. 8.

Evolutionary search is based on a set $\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\mu\}$ of parental and a set $\mathcal{P}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_\lambda\}$ of offspring candidate solutions, also called individuals. The individuals are iteratively subject to random changes and selection of the best solutions. Algorithm 1 shows the pseudocode of a general evolutionary algorithm. The optimization process consists of three main steps:

1. The recombination operator selects ρ parents and combines their parts to λ new solutions.
2. The mutation operator adds random changes (e.g. noise) to the preliminary candidate solution. The quality of the individuals in solving the optimization problem

Algorithm 1 Evolutionary Algorithm

```
1: initialize solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\mu \in \mathcal{P}$
2: repeat
3:   for $i = 1$ to $\lambda$ do
4:     select $\rho$ parents from $\mathcal{P}$
5:     create $\mathbf{x}_i$ by recombination
6:     mutate $\mathbf{x}_i$
7:     evaluate $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
8:     add $\mathbf{x}_i$ to $\mathcal{P}'$
9:   end for
10:  select $\mu$ parents from $\mathcal{P}' \rightarrow \mathcal{P}$
11: until termination condition
```

is called fitness. The fitness of the new offspring solution is evaluated on fitness function f. All individuals of a generation are put into offspring population $\mathcal{P}'$.

3. Then, μ individuals are selected and constitute the novel parental population $\mathcal{P}$ of the following generation.

The process is repeated until a termination condition is reached. Typical termination conditions are defined via fitness values or via an upper bound on the number of generations.

In the following, we will give a short survey of evolutionary operators and go deeper into evolution strategies that have proven well in practical optimization scenarios. The evolution strategy operators intermediate and dominant recombination as well as Gaussian mutation are introduced.

2.3 Recombination

In biological systems, recombination, also known as crossover, mixes the genetic material of two parents. Most evolutionary algorithms also make use of a recombination operator and combine the information of two or more individuals $\mathbf{x}_1, \ldots, \mathbf{x}_\rho$ to a new offspring solution. Hence, the offspring carries parts of the genetic material of its parents. Many recombination operators are restricted to two parents, but also multi-parent recombination variants have been proposed in the past that combine information of ρ parents. The use of recombination is discussed controversially within the building block hypothesis by Goldberg [16, 17]. The building block hypothesis assumes that good substrings of the solutions called building blocks of different parents are combined, and their number increases. The *good genes* are spread over the population in the course of the evolutionary process.

Typical recombination operators for continuous representations are dominant and intermediate recombination. Dominant recombination randomly combines the genes of all parents. Dominant recombination with ρ parents $(\mathbf{x})_1, \ldots, (\mathbf{x})_\rho \in \mathbb{R}^N$ creates the offspring vector $\mathbf{x}' = (x'_1, \ldots, x'_N)^T$ by randomly choosing the i-th component

$$x'_i = (x_i)_k, \quad k \in \text{random}\{1, \ldots, \rho\}. \tag{2.1}$$

Intermediate recombination is appropriate for numerical solution spaces. Given ρ parents $\mathbf{x}_1, \ldots, \mathbf{x}_\rho$ each component of the offspring vector $\mathbf{x}'$ is the arithmetic mean of the components of all ρ parents

$$x_i' = \frac{1}{\rho} \sum_{k=1}^{\rho} (x_i)_k. \tag{2.2}$$

The characteristics of offspring solutions lie between their parents. Integer representations may require rounding procedures for generating valid solutions.

2.4 Mutation

Mutation is the second main source of evolutionary changes. According to Beyer and Schwefel [3], a mutation operator is supposed to fulfill three conditions. First, from each point in the solution space each other point must be reachable. Second, in unconstrained solution spaces a bias is disadvantageous, because the direction to the optimum is unknown, and third, the mutation strength should be adjustable, in order to adapt exploration and exploitation to local solution space conditions.

In the following, we concentrate on the famous Gaussian mutation operator for optimization in $\mathbb{R}^N$. Solutions are vectors of real values $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$. Random numbers based on the Gaussian distribution $\mathcal{N}(0, 1)$ fulfill these conditions in continuous domains.[1] With the Gaussian distribution, many natural and artificial processes can be described. The idea is to mutate each individual applying the mutation operator

$$\mathbf{x}' = \mathbf{x} + \mathbf{z}, \tag{2.3}$$

with a mutation vector $\mathbf{z} \in \mathbb{R}^N$ based on sampling from the Gaussian distribution

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) = (\mathcal{N}(0, \sigma^2), \ldots, \mathcal{N}(0, \sigma^2))^T \sim \sigma\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.4}$$

with identity matrix $\mathbf{I}$. The standard deviation σ plays the role of the mutation strength and is also known as step size. The isotropic Gaussian mutation with only one step size uses the same standard deviation for each component x_i. Of course, the step size σ can be kept constant, but the convergence to the optimum can be improved by adapting σ according to local solution space characteristics. In case of high success rates, i.e., a large number of offspring solutions being better than their parents, big step sizes are advantageous, in order to explore the solution space as fast as possible. This is often reasonable at the beginning of the search. In case of low success rates, smaller step sizes are appropriate. This is often adequate in later phases of the search during convergence to the optimum, i.e., when approximating solutions should not

[1] $\mathcal{N}(m, \sigma^2)$ represents a randomly drawn Gaussian distributed number with expectation value m and standard deviation σ.

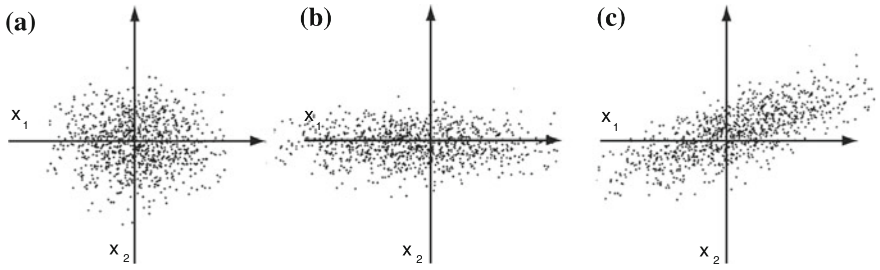


Fig. 2.1 Gaussian mutation: **a** isotropic Gaussian mutation employs one step size σ for each dimension, **b** multivariate Gaussian mutation allows independent step sizes in each dimension, and **c** correlated mutation allows a rotation of the mutation ellipsoid, (**a**) isotropic, (**b**) multivariate, (**c**) correlated

be destroyed. An example for an adaptive control of step sizes is the $1/5$-th success rule by Rechenberg [4] that increases the step sizes, if the success rate is over $1/5$-th, and decreases it, if the success rate is lower.

Isotropic Gaussian mutation can be extended to multivariate Gaussian mutation by allowing independent scalings of the components

$$\mathcal{N}(\mathbf{0}, \mathbf{D}^2) = (\mathcal{N}(0, \sigma_1^2), \ldots, \mathcal{N}(0, \sigma_N^2))^T \sim \mathbf{D}\mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.5}$$

This multivariate mutation employs N degrees of freedom that are saved in a diagonal matrix $\mathbf{D} = \text{diag}(\sigma_1, \ldots, \sigma_N)$ corresponding to a step size vector for the independent scalings.

Figure 2.1 illustrates the differences between (a) isotropic Gaussian mutation and (b) multivariate Gaussian mutation. The multivariate variant allows the development of a Gaussian ellipsoid that flexibly adapts to local solution space characteristics. Even more flexibility, i.e., $N(N-1)/2$ degrees of freedom, allows the correlated mutation presented by Schwefel [18]

$$\mathcal{N}(\mathbf{0}, \mathbf{C}) = \sqrt{\mathbf{C}}\mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.6}$$

with covariance matrix $\mathbf{C}$, which contains the covariances describing the multivariate normal distribution. The components are correlated, see Fig. 2.1c. The *square root*, or Cholesky decomposition, $\sqrt{\mathbf{C}}$ of the covariance matrix $\mathbf{C}$ corresponds to a rotation matrix for the mutation ellipsoid axes. The question arises, how to control the mutation ellipsoid rotation. Instead of a rotation matrix $N(N-1)/2$ angles can be used. In practical optimization, these angles are often controlled self-adaptively [19]. Also the CMA-ES and variants [7, 20] are based on an automatic alignment of the coordinate system (cf. Sect. 2.7).

A step towards the acceleration of the step size control is σ-self-adaptation. Before the application of the mutation operator (cf. Eqs. 2.3 and 2.4), the log-normal mutation operator for step sizes σ and step size vectors $(\sigma_1, \ldots, \sigma_N)^T$ is applied. The log-normal mutation operator has been proposed by Schwefel [18] and has become famous for self-adaptation in continuous solution spaces. It is defined as

$$\sigma' = \sigma \cdot e^{\tau \cdot \mathcal{N}(0,1)}. \tag{2.7}$$

The problem-dependent learning rate τ has to be chosen adequately. For the mutation strengths of evolution strategies on the continuous *Sphere* model, theoretical investigations [3] lead to the optimal setting

$$\tau = \frac{1}{\sqrt{N}}, \tag{2.8}$$

which may not be optimal for other problems, and further parameter tuning is recommended. Strategy parameter σ cannot become negative and scales logarithmically between values close to 0 and infinity.[2] A more flexible approach is to mutate each of the N dimensions independently

$$\sigma' = e^{(\tau_0 \mathcal{N}(0,1))} \cdot \left(\sigma_1 e^{(\tau_1 \mathcal{N}(0,1)}, \ldots, \sigma_N e^{(\tau_1 \mathcal{N}(0,1)}\right), \tag{2.9}$$

with

$$\tau_0 = \frac{c}{\sqrt{2 \cdot N}}, \tag{2.10}$$

and

$$\tau_1 = \frac{c}{\sqrt{2\sqrt{N}}}. \tag{2.11}$$

Setting parameter $c = 1$ is a recommendable choice. *Kursawe* [21] analyzed parameters τ_0 and τ_1 using a nested evolution strategy on various test problems. His analysis shows that the choice of mutation parameters is problem-depended, and general recommendations are difficult to give.

The EA performs the search in two spaces: the objective and the strategy parameter space. Strategy parameters influence the genetic operators of the objective variable space, in this case the step sizes of the Gaussian mutation. The optimal settings may vary depending on the location of the solution in the fitness landscape. Only the objective variables define the solution and have an impact on the fitness. Strategy parameters have to take part in the evolutionary process to evolve them dynamically during the optimization process.

2.5 Selection

The counterpart of the variation operators mutation and recombination is selection. Selection gives the evolutionary search a direction. Based on their fitness, a subset of the population is selected, while the worst individuals are rejected. In the evolutionary framework, the selection operator can be employed at two steps. Mating selection

[2] i.e., high values w.r.t. the data structure.

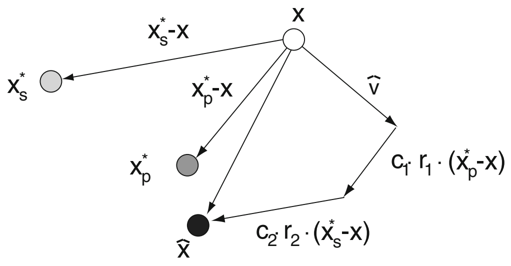
selects individuals for the recombination operator. In nature, the attraction of sexual partners as well as cultural aspects influence the mating selection process. The second famous selection operator is survivor selection corresponding to the Darwinian principle of *survival of the fittest*. Only the individuals selected by survivor selection are allowed to inherit their genetic material to the following generation. The probability of a solution to be selected is also known as selection pressure.

Evolution strategies usually do not employ a competitive selection operator for mating selection. Instead, parental solutions are randomly drawn from the set of candidate solutions. But for survivor selection, the elitist selection strategies comma and plus selection are used. They choose the μ-best solutions as basis for the parental population of the following generation. Both operators, plus and comma selection, can easily be implemented by sorting the population w.r.t. the individuals' fitness. Plus selection selects the μ-best solutions from the union $\mathcal{P} \cup \mathcal{P}'$ of the last parental population $\mathcal{P}$ and the current offspring population $\mathcal{P}'$, and is denoted by $(\mu + \lambda)$-ES. In contrast, comma selection in a (μ, λ)-ES selects exclusively from the offspring population, neglecting the parental population, even if the parents have a superior fitness. Forgetting superior solutions may seem to be disadvantageous. But potentially good solutions may turn out to be local optima, and the evolutionary process may fail to leave them without the ability to forget.

2.6 Particle Swarm Optimization

Similar to evolutionary algorithms, PSO is a population approach with stochastic components. Introduced by Kennedy and Eberhart [13], it is inspired by the movement of natural swarms and flocks. The algorithm utilizes particles with a position $\mathbf{x}$ that corresponds to the optimization variables and a velocity $\mathbf{v}$, which is similar to the mutation strengths in evolutionary computation. The principle of PSO is based on the idea that the particles move in solution space influencing each other with stochastic changes, while previous successful solutions act as attractors. Figure 2.2 illustrates the PSO concept in $N = 2$ dimensions, while Algorithm 2 shows the PSO algorithm in pseudocode.

In each iteration the position of particle $\mathbf{x}$ is updated by adding a velocity $\hat{\mathbf{v}}$



x
$x_s^* - x$
x_s^*
$x_p^* - x$
$\hat{v}$
x_p^*
$c_1 \cdot r_1 \cdot (x_p^* - x)$
$c_2 \cdot r_2 \cdot (x_s^* - x)$
$\hat{x}$

Fig. 2.2 Illustration of PSO concept

Algorithm 2 Particle Swarm Optimization Algorithm

```
1: initialize parameters, and particles
2: repeat
3:   for $i = 1$ to $\kappa$ do
4:     compute $\mathbf{x}_p^*$, and $\mathbf{x}_s^*$
5:     update velocity $\hat{\mathbf{v}}$
6:     update position $\hat{\mathbf{x}}$
7:     compute fitness $f(\hat{\mathbf{x}})$
8:   end for
9: until termination condition
```

$$\hat{\mathbf{x}} = \mathbf{x} + \hat{\mathbf{v}}, \tag{2.12}$$

which is updated as follows

$$\hat{\mathbf{v}} = \mathbf{v} + c_1 r_1 (\mathbf{x}_p^* - \mathbf{x}) + c_2 r_2 (\mathbf{x}_s^* - \mathbf{x}), \tag{2.13}$$

where $\mathbf{x}_p^*$, and $\mathbf{x}_s^*$ denote the best previous positions of the particle, and of the swarm, respectively. The weights $c_1, c_2 \in [0, 1]$ are acceleration coefficients that determine the bias of the particle towards its own, and the swarm history. The recommendation given by Kennedy and Eberhart is to set both parameters to $c_1 = c_2 = 0.5$ [13]. The random components r_1, and r_2 are uniformly drawn from the interval [0, 1], and can be used to control exploitation and exploration of the solution space.

2.7 Covariance Matrix Adaptation Evolution Strategies

In the following, we introduce an algorithm from the family of covariance matrix adaptation evolution strategies (CMA-ES). The covariance matrix self-adaptation evolution strategy (CMSA-ES) by Beyer and Sendhoff [20] is the historically latest covariance matrix adaptation-based strategy, but is a variant that reflects well the main idea of the family of CMA-ES. The basic idea is to align the coordinate system for the mutation operator to the distribution of the selected solutions in each generation. The aligned coordinate system guarantees that mutations in the following generation are similar to the best of the previous generation. The CMSA-ES is a variant of the famous CMA-ES by Hansen and Ostermeier [7] with an emphasis on self-adaptation.

The CMSA-ES is based on a self-adaptive step control of step sizes, similar to the $(\mu \overset{+}{,} \lambda)$-ES introduced in the previous section. After initialization, λ candidate solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ are generated. With the help of the global self-adaptive, N-dimensional step size $\hat{\sigma} = \frac{1}{\mu} \sum_{j=1}^{\mu} \sigma_{j:\lambda}$, which is the arithmetic mean of the step sizes from the μ-best solutions of λ offspring solutions[3] of the previous generation, each individual gets a log-normally mutated step size

[3] The index j denotes the index of the j-th ranked individual of the λ offspring individuals w.r.t. an increasing sorting based on fitness $f(\mathbf{x}_j)$.

$$\sigma_j = \hat{\sigma} \cdot e^{\tau_\sigma \mathcal{N}(0,1)}. \tag{2.14}$$

The main idea of the approach is to align the coordinate system by changing the coordinates $\mathbf{x}_j$ with the help of the current mean $\hat{\mathbf{x}}$ of the population and a covariance matrix $\mathbf{C}$ based on the best solutions and the past optimization process. From $\mathbf{C}$ the correlated random directions $\mathbf{s}_j$ are generated by multiplication of the Cholesky decomposition $\sqrt{\mathbf{C}}$ with the standard normal vector $\mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{s}_j \sim \sqrt{\mathbf{C}}\mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.15}$$

This random direction is scaled in length w.r.t. the self-adaptive step size σ_j

$$\mathbf{z}_j = \sigma_j \mathbf{s}_j. \tag{2.16}$$

The resulting vector $\mathbf{z}_j$ is added to the global parent $\hat{\mathbf{x}}$, i.e.

$$\mathbf{x}_j = \hat{\mathbf{x}} + \mathbf{z}_j. \tag{2.17}$$

Finally, fitness $f_j = f(\mathbf{x}_j)$ of solution $\mathbf{x}_j$ is evaluated. When λ offspring solutions have been generated, the μ-best solutions are selected and their components $\mathbf{z}_j$ and σ_j are recombined. Beyer and Sendhoff [20] apply global recombination, i.e., the arithmetic mean of each parameter is computed. The outer product $\mathbf{s}\mathbf{s}^T$ of the search directions is an estimation of the covariance matrix of the best mutations and is computed for each of the μ-best solutions and averaged afterwards

$$\mathbf{S} = \frac{1}{\mu} \sum_{j=1}^{\mu} \mathbf{s}_{j:\lambda} \mathbf{s}_{j:\lambda}^T. \tag{2.18}$$

Last, the covariance matrix $\mathbf{C}$ is updated based on the current covariance matrix and the new estimation $\mathbf{S}$. The covariance matrix update

$$\mathbf{C} = \left(1 - \frac{1}{\tau_c}\right)\mathbf{C} + \frac{1}{\tau_c}\mathbf{S} \tag{2.19}$$

is a composition of the last covariance matrix $\mathbf{C}$ and the outer product of the search direction of the μ-best solutions balanced by learning parameter τ_c. Adapted in such a kind of way, sampling from a Gaussian distribution based on $\mathbf{C}$ increases the likelihood of successful steps. Beyer and Sendhoff recommend to set

$$\tau_c = \frac{N(N+1)}{2\mu} \tag{2.20}$$

for the learning parameter. All steps are repeated until a termination condition is satisfied. The CMSA-ES combines the self-adaptive step size control with a simul-

Algorithm 3 CMSA-ES

```
1: initialize solution $\hat{\mathbf{x}}$
2: repeat
3:   for $i = 1$ to $\lambda$ do
4:     $\sigma_j \sim \hat{\sigma} \cdot e^{\tau_\sigma \mathcal{N}(0,1)}$
5:     $\mathbf{s}_j \sim \sqrt{\mathbf{C}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
6:     $\mathbf{z}_j = \sigma_j \mathbf{s}_j$
7:     $\mathbf{x}_j = \hat{\mathbf{x}} + \mathbf{z}_j$
8:     $f_j = f(\mathbf{x}_j)$
9:   end for
10:  sort population w.r.t. fitness $f_{j:\lambda}$
11:  $\hat{\mathbf{z}} = \frac{1}{\mu} \sum_{j=1}^{\mu} \mathbf{z}_{j:\lambda}$
12:  $\hat{\sigma} = \frac{1}{\mu} \sum_{j=1}^{\mu} \sigma_{j:\lambda}$
13:  $\mathbf{S} = \frac{1}{\mu} \sum_{j=1}^{\mu} \mathbf{s}_{j:\lambda} \mathbf{s}_{j:\lambda}^T$
14:  $\hat{\mathbf{x}} = \hat{\mathbf{x}} + \hat{\mathbf{z}}$
15:  $\mathbf{C} = (1 - \frac{1}{\tau_c})\mathbf{C} + \frac{1}{\tau_c}\mathbf{S}$
16: until termination condition
```

taneous update of the covariance matrix. Algorithm 3 shows the pseudocode of the CMSA-ES. Initially, the covariance matrix $\mathbf{C}$ is chosen as the identity matrix $\mathbf{C} = \mathbf{I}$. The learning parameter τ_σ defines the mutation strength of the step sizes σ_j. For the *Sphere* problem, the optimal learning parameter is $\tau_\sigma = \frac{1}{\sqrt{2 \cdot N}}$ [3].

In the following, we present an experimental analysis of the CMSA-ES concentrating on typical test problems known in literature [22] (cf. Appendix A). We use the following performance measure. The experimental results show the number of fitness function evaluations until the optimum is reached with accuracy f_{stop}, i.e., if the difference between the best achieved fitness $f(\mathbf{x}')$ of the algorithm and fitness $f(\mathbf{x}^*)$ of the known optimum $\mathbf{x}^*$ is smaller than f_{stop}, i.e.,

$$|f(\mathbf{x}') - f(\mathbf{x}^*)| \leq f_{\text{stop}}. \tag{2.21}$$

This performance measure is focused on the convergence abilities of the approach. The figures of Table 2.1 show the best, median, worst, and mean (with standard deviation) number of generations until the termination condition has been reached. The termination condition is set to $f_{\text{stop}} = 10^{-10}$.

The results confirm that the CMSA-ES is a strong method for derivative-free multimodal optimization. It is able to find the optima of all test problems. In case of the unimodal problems *Sphere* and *Doublesum*, no restarts have been necessary. The performance comparison in the later chapters will allow a detailed interpretation.

Table 2.1 Experimental analysis of CMSA-ES with restarts

	Best	Median	Worst	Mean	Dev
$N = 10$					
f_{Sp}	2,120	2,195	2,350	2,204.5	7.0e1
f_{Dou}	2,280	2,355	2,490	2,358.3	6.2e1
f_{Ros}	7,060	10,550	18,080	11,292.0	4.1e3
f_{Ras}	36,360	90,540	203,120	103,456.0	5.7e4
f_{Gri}	2,150	4,375	13,090	5,579.7	4.1e3
f_{Kur}	10,780	21,960	81,370	29,670.9	22.0e3
$N = 30$					
f_{Sp}	5,684	5,880	6,118	5,896.8	1.4e2
f_{Dou}	7,770	8,092	8,302	8,075.2	1.6e2
f_{Ros}	45,976	51,681	109,984	58,595.6	1.9e4
f_{Ras}	360,990	699,846	721,224	576,511.6	1.7e5
f_{Gri}	6,370	6,755	17,374	8,764	4.4e3
f_{Kur}	55,244	89,138	138,670	93,518.6	37.4e3

2.8 Conclusions

Evolution strategies, in particular the covariance matrix adaptation variants, belong to the most successful evolutionary optimization algorithms for solving black box optimization problems. If no derivatives are given and no assumptions about the fitness function are available, the application of evolutionary algorithms is a recommendable undertaking. Theoretical results and a huge variety of applications have proven their success in the past. But the success of evolutionary search also depends on proper parameter settings before and during the search. We will concentrate on the parameter control problem in the next chapter. We have already introduced σ-self-adaptation as parameter control techniques for steps sizes in evolution strategies, which is based on evolutionary search in the space of step sizes. This mechanism has significantly contributed to the success of evolutionary optimization methods.

References

1. O. Kramer, D. E. Ciaurri, S. Koziel, Derivative-free optimization, in *Computational Optimization and Applications in Engineering and Industry, Studies in Computational Intelligence* (Springer, New York, 2011). pp. 61–83
2. J. Nocedal, S. J. Wright, *Numerical Optimization* (Springer, New York, 2000)
3. H.-G. Beyer, H.-P. Schwefel, Evolution strategies—a comprehensive introduction. Nat. Comput. **1**, 3–52 (2002)
4. I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* (Frommann-Holzboog, Stuttgart, 1973)
5. H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie* (Birkhäuser, Basel, 1977)

6. J.H. Holland, *Adaptation in Natural and Artificial Systems*, 1st edn (MIT Press, Cambridge, 1992)
7. A. Ostermeier, A. Gawelczyk, N. Hansen, A derandomized approach to self adaptation of evolution strategies. Evol. Comput. **2**(4), 369–380 (1994)
8. K. Deb, A. Anand, D. Joshi, A computationally efficient evolutionary algorithm for real-parameter optimization. Evol. Comput. **10**(4), 371–395 (2002)
9. F. Herrera, M. Lozan, J.L. Verdegay, Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artif. Intell. Rev. **12**, 265–319 (1998)
10. F. Herrera, M. Lozano, Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. Appl. Intell. **13**(3), 187–204 (2000)
11. K.V. Price, R.M. Storn, J.A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization* (Springer, Natural Computing Series, New York, 2005)
12. A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
13. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of IEEE International Conference on Neural Networks* (1995). pp. 1942–1948
14. Y. Shi, R. Eberhart, A modified particle swarm optimizer, in *Proceedings of the International Conference on Evolutionary Computation* (1998). pp. 69–73
15. J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
16. D. Goldberg, *Genetic Algorithms in Search* (Optimization and Machine Learning. Addison-Wesley, Reading, 1989)
17. J.H. Holland, *Hidden Order: How Adaptation Builds Complexity* (Addison-Wesley, Reading, 1995)
18. H.-P. Schwefel, *Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit* (Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik, TU Berlin, July 1974)
19. O. Kramer, *Self-Adaptive Heuristics for Evolutionary Computation, Studies in Computational Intelligence* (Springer, Heidelberg, 2008)
20. H.-G. Beyer, B. Sendhoff, Covariance matrix adaptation revisited—the CMSA evolution strategy, in *Proceedings of the 10th Conference on Parallel Problem Solving from Nature* (*PPSN*, 2008) pp. 123–132
21. F. Kursawe, Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien—Metastrategien. PhD thesis, University of Dortmund, 1999
22. P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.p. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC special Session on real-parameter optimization, Technical report, Nanyang Technological University, 2005