Chapter 4

Evolutionary Algorithms and Evolution Strategies

The paradigm of evolutionary computation is derived from the model of organic evolution and refers to the application of the Darwinian principles of evolution for computational purposes. The term *Evolutionary Algorithm* refers to an algorithmic method that adopts the paradigm of evolutionary computation for solving optimization problems. Within these methods, a *population* of candidate solutions (*individuals*) repeatedly undergoes the processes of reproduction and selection, with the *fitness* of each candidate solution being expressed in terms of its quality with respect to the given optimization problem. Hence, the basic elements of natural evolution are used to breed populations of (near-)optimal solutions.

The main components of an Evolutionary Algorithm are summarized in Section 4.1. For a thorough introduction to Evolutionary Algorithms, the reader is referred to, e.g., [Bäc96] and [ES03]. Section 4.2 focuses on Evolution Strategies and introduces the two algorithmic techniques that are the main focus of this work; the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES. Section 4.3 closes with a summary and discussion.

4.1 Evolutionary Algorithms

Figure 4.1 depicts the general evolution cycle of an Evolutionary Algorithm as a flowchart. This flowchart is one of the possible variants. It considers a population of candidate solutions or individuals that, after being initialized in some fashion, enters an evolution loop. A subset of this population is selected (based on fitness) as *parents* for creating the *offspring* of the next *generation*. The parents are then used for generating a set of offspring by *recombination* of two or more of the parents and *mutation* of the recombined individual. Thereafter, the population of the next generation is selected either from the offspring and the parents (*elitism*) or solely from the offspring. This loop is repeated until a termination criterion is met.

Representation: Each individual represents a candidate solution for the optimization problem at hand and should be modeled in some appropriate/convenient way in order to be used within

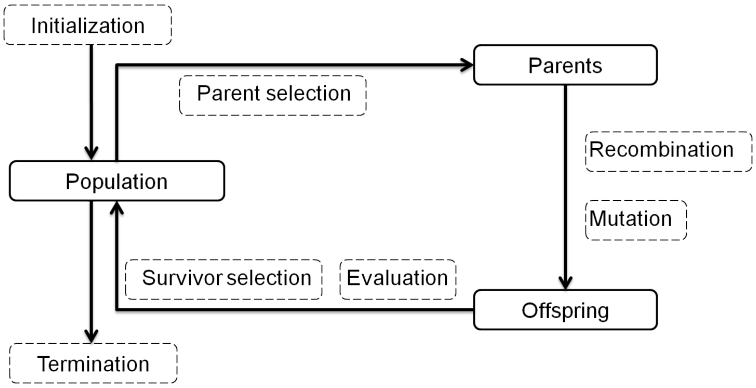

Figure 4.1: The iteration cycle of an Evolutionary Algorithm.

the context of Evolutionary Algorithms. For some problems, the representation follows naturally from the description of the search space. For other problems, an abstract representation has to be designed, making a genotype/phenotype distinction. In the latter case, also a genotype to phenotype decoding mechanism has to be designed. Two key points in choosing an appropriate representation are: 1) It should be possible for each element of the search space to be modeled (preferable uniquely) according to the chosen representation. 2) The representation should allow for the implementation of genetic operators such as mutation and recombination.

Initialization: The initial population can be based on known good solutions that serve as starting point for the evolutionary search, or can be generated randomly. When generating the initial population randomly, it should be well-spread over the search space, which yields a higher chance of identifying promising regions in the search space. For most representations, the initialization procedure is fairly straightforward (e.g., using Latin Hypercube sampling in real-parameter search spaces).

Evaluation: The evaluation component uses the objective and constraint functions of the optimization problem to assign a quality score or fitness to each individual. Individuals with a higher fitness will have a higher probability of surviving and passing on their genetic material (i.e., the candidate solution that they represent) to future generations.

Selection: There are two types of selection: *parental selection* (or *mating selection*) and *survivor selection* (or *environmental selection*). Parental selection is a stochastic selection type that selects the parents that are used for the recombination of a new offspring (i.e., for the generation of each offspring). In this selection type, the fitter individuals have a higher probability to be selected as parent for recombination. The latter, survivor selection, is a deterministic selection type that is applied at an earlier stage. It selects the μ fittest individuals (the survivors) either out of the λ offspring, or, if *elitism* is used, out of the λ offspring and the μ old parents. This selection type is commonly referred to as $(\mu^{+}_{,}\lambda)$-selection, with $(\mu{+}\lambda)$ denoting elitist selection, and (μ, λ) denoting non-elitist selection.

Mutation and recombination: Mutation and recombination are the two types of *genetic operators* or *variation operators*. Mutation operators add small perturbations to the (genotype representations of the) individuals in the population. Recombination operators recombine two or more individuals in the population into a new individual by means of crossover (of the chromosomes). The choice of the genetic operators depends on the chosen representation of individuals. An important criterion for the genetic operators is that it should be possible to get from any solution in the search space to any other solution in the search space by applying the genetic operators a finite number of times. Besides that, the genetic operators should be well-defined, such that applying the operators to any solution (set of solutions) yields a valid solution that is also in the search space. Lastly, the genetic operators should be unbiased.

Termination: The termination condition can depend on the available computation time, the available number of evaluations/generations, or on convergence criteria such as a pre-defined target fitness that is to be reached. After termination, the best solution(s) of the final population or the best solution(s) found throughout the evolution loop can be considered as (a) good solution(s) for the optimization problem. That is, provided the evolution process is granted enough time.

4.2 Evolution Strategies

Evolution Strategies form a special branch of Evolutionary Algorithms, specifically designed for real-parameter optimization problems as described by Definition 2.1.11 on page 16. They have been proposed originally by Rechenberg [Rec73] and Schwefel [Sch77]. Over the years, the class of Evolution Strategies has broadened into many algorithmic variants, making it hard to pin down the canonical Evolution Strategy. In this section we will briefly summarize the basic concepts and the most important Evolution Strategy variants with their respective algorithmic descriptions.

4.2.1 The $(1+1)$-Evolution Strategy

The simplest Evolution Strategy is the $(1+1)$-Evolution Strategy (one parent, one offspring), presented by Rechenberg [Rec73]. Algorithm 4.1 describes the working mechanism of this simple Evolution Strategy.

The $(1+1)$-ES starts with a random solution $\mathbf{x}_\mathrm{p}$ drawn uniform randomly from the search space $[\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u}]$ (denoted $\mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$). This solution, the parent individual, is evaluated and the algorithm enters the evolution cycle. In each generation, one offspring $\mathbf{x}_\mathrm{o}$ is created from the parent by adding a small random vector to a copy of the parent. The perturbation, or mutation, is drawn from an uncorrelated multivariate Gaussian distribution:

$$\mathbf{x}_\mathrm{o} = \mathbf{x}_\mathrm{p} + \sigma\mathbf{z} \ , \ \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.1}$$

Algorithm 4.1: The $(1+1)$-Evolution Strategy

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{p}$ with objective function value f_p

Set parameters: $c \leftarrow 0.85$, $G \leftarrow \max(5, n)$

Initialize: $g \leftarrow 0$, $G_\mathrm{s} \leftarrow 0$, $\sigma \leftarrow \frac{||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||}{3\sqrt{n}}$, $\mathbf{x}_\mathrm{p} \leftarrow \mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$, $f_\mathrm{p} \leftarrow f(\mathbf{x}_\mathrm{p})$

while not terminate **do**

  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

  $\mathbf{x}_\mathrm{o} \leftarrow \mathbf{x}_\mathrm{p} + \sigma\mathbf{z}$

  $f_\mathrm{o} \leftarrow f(\mathbf{x}_\mathrm{o})$

  if $f_\mathrm{o} \leq f_\mathrm{p}$ **then**

    $\mathbf{x}_\mathrm{p} \leftarrow \mathbf{x}_\mathrm{o}$

    $f_\mathrm{p} \leftarrow f_\mathrm{o}$

    $G_\mathrm{s} \leftarrow G_\mathrm{s} + 1$

  end if

  $g \leftarrow g + 1$

  if $g \bmod G = 0$ **then**

    $p_\mathrm{s} \leftarrow G_\mathrm{s}/G$

    $$\sigma \leftarrow \begin{cases} \sigma/c & , \text{if } p_\mathrm{s} > 1/5 \\ \sigma \cdot c & , \text{if } p_\mathrm{s} < 1/5 \\ \sigma & , \text{otherwise} \end{cases}$$

    $G_\mathrm{s} \leftarrow 0$

  end if

end while

return $(\mathbf{x}_\mathrm{p}, f_\mathrm{p})$

The newly generated offspring is evaluated and replaces the parent if it has a better fitness. This loop is repeated until the termination criterion is met. After termination, the best solution (i.e., the parent after the last iteration) is returned together with its fitness value.

The magnitude of the random perturbation added to the offspring is determined by σ, which is the so-called *stepsize parameter*. It is updated every G iterations based on the success rate p_s, which is the ratio of the mutations that generated an offspring fitter than the parent. The update of σ follows the so-called $1/5$*th-success rule* [Rec73]; if the success rate is higher than $1/5$th, the stepsize should be increased, if the success rate is lower than $1/5$th, the stepsize should be decreased, and if the success rate is equal to $1/5$th, it remains unchanged. For increasing and decreasing the stepsize, a constant multiplication factor $0.817 \leq c < 1$ is used.

For the parameter c, a reasonable setting is $c = 0.85$ (see [Bäc96]). For the parameter G, a reasonable setting is $G = n$ for $n \geq 5$. The initial stepsize σ used here is proportional to the expected distance to the optimum of the initial parent, which according to Schwefel [Sch77] is a reasonable initial stepsize. In this work we adopt an initialization of $\sigma = ||\mathbf{x}_u - \mathbf{x}_l||/(3\sqrt{n})$. This is based on the fact that the expected length of a mutation step is $\mathbf{E}[\mathbf{z}] = \sigma\sqrt{n}$ and that the average distance between two random points in an n-dimensional cube is proportional to $\sqrt{n}$ [BP09] (we take $||\mathbf{x}_u - \mathbf{x}_l||/3$ as an approximation), hence, the initial mutation step is taken to be proportional to the average distance of the initial solution to the optimum.

4.2.2 The $(\mu/\rho^+_,\lambda)$-SA-Evolution Strategy

The $(\mu/\rho^+_,\lambda)$-SA-ES, originally proposed by Schwefel [Sch77], is a population based extension of the $(1+1)$-ES. Instead of generating only one offspring from one parent, λ offspring are generated from μ parents, based on both recombination and mutation. Another difference as compared to the $(1+1)$-ES is the adaptation of the control parameters of the mutation operator.

Algorithm 4.2 shows the general outline of a $(\mu/\rho^+_,\lambda)$-SA-ES. Individuals are of the form $\mathbf{a} = (\mathbf{x}, \mathbf{s}, f)$, where $\mathbf{x} \in \mathbb{R}^n$ is a vector of *object variables* (i.e., the candidate solution represented by the individual), $\mathbf{s}$ is a set of *endogenous strategy parameters* (or just *strategy parameters*), and f holds the fitness value. The strategy parameters are control parameters for the mutation operator and are evolved together with the object variables. By including the strategy parameters in the evolution process, these parameters do not have to be set externally, but evolution itself adapts them to appropriate settings. Co-evolution of internal strategy parameters is called *self-adaptation* (SA).

The algorithm starts with an initial *parent population* P_p consisting of μ parents, initialized in some fashion. For each generation, λ offspring are generated from the parent population P_p, evaluated, and added to the *offspring population* P_o. After generating the offspring, μ individuals are selected either solely from the offspring population (*comma-selection*) or from the union of the parent population and the offspring population (*plus-selection*) to form the parent population of the next generation. At the end of each generation the generation counter

g is increased and the best solution of the new parent population is stored, to be returned after termination. This procedure is repeated until the termination criterion is met.

For the generation of each offspring, a set $R \subseteq P_{\mathrm{p}}$ of ρ parents is selected randomly from the parent population for generating the new offspring (*marriage*). The strategy parameters and object variables of the selected parents are recombined (*recombine_s* and *recombine_x*) to form recombinants $\mathbf{s}$ and $\mathbf{x}$. These recombinants are thereafter mutated (*mutate_s* and *mutate_x*) and evaluated to form the new offspring. Note that first the strategy parameters, then the object variables should be mutated (according to the new strategy parameters) in order to achieve co-evolution of the strategy parameters.

Marriage: The *marriage* operation selects a random subset of ρ parents for recombination. The parameter ρ is called the *mixing number*. Within Evolution Strategies, recombination is commonly done either with $\rho = 2$ parents or with $\rho = \mu$ parents (*global recombination*).

Recombination: There are two different types of recombination: *discrete* and *intermediate*. Recombination is used for both the object variables and the strategy parameters. With discrete recombination, for each element of the recombinant it is decided uniform randomly from which of the parents the corresponding element should be copied. That is, given the ρ parental vectors $\{\mathbf{r}_1^{\mathrm{p}}, \ldots, \mathbf{r}_\rho^{\mathrm{p}}\}$, the ith of the new recombinant $\mathbf{r}^{\mathrm{o}}$ is constructed by

$$(\mathbf{r}^{\mathrm{o}})_i = \left(\mathbf{r}_{m_i}^{\mathrm{p}}\right)_i \ , \ m_i = \mathrm{rand}\{1, \ldots, \rho\}. \tag{4.2}$$

Here, the recombinant $\mathbf{r}^{\mathrm{o}}$ can be either the recombinant of the object variables $\mathbf{x}$ or of the strategy parameters $\mathbf{s}$. With intermediate recombination, the value of each element of the offspring's object variables or strategy parameters is set to the average over all parents. That is,

$$(\mathbf{r}^{\mathrm{o}})_i = \frac{1}{\rho} \sum_{j=1}^{\rho} \left(\mathbf{r}_j^{\mathrm{p}}\right)_i . \tag{4.3}$$

Figure 4.2 illustrates the working mechanism of the two different recombination schemes for recombination of two parents.

Within Evolution Strategies, it is not uncommon that different recombination types are used for the object variables and the strategy parameters. Standard practice is to use discrete recombination for the object variables, and intermediate recombination for the strategy parameters. Regarding notation, the recombination type is sometimes included in the denotation of particular Evolution Strategies by means of two subscripts on the mixing number ρ (D for discrete and I for intermediate). The first subscript denotes the recombination type for the object variables and the second denotes the recombination type for the strategy parameters. For example, a $(\mu/\rho_{DI}^{+},\lambda)$-ES denotes a $(\mu/\rho^{+},\lambda)$-ES with discrete recombination of the object variables and intermediate recombination of the strategy parameters.

Mutation: As with the $(1{+}1)$-ES, mutation works by adding a random vector generated from a multivariate Gaussian distribution, which in the simplest case is uncorrelated and isotropic.

Algorithm 4.2: General Outline of a $(\mu/\rho^{+},\lambda)$-SA-ES

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{opt}$ with objective function value f_opt

Initialize: $g \leftarrow 0$, $P_\mathrm{p} \leftarrow$ initialize $(\{\mathbf{a}_k = (\mathbf{x}_k, \mathbf{s}_k, f_k), k = 1, \ldots, \mu\})$
while not terminate **do**
  $P_\mathrm{o} \leftarrow \emptyset$
  for $k = 1 \to \lambda$ **do**
    $R_k \leftarrow$ marriage (P_p, ρ)
    $\mathbf{s}_k \leftarrow$ recombine_s (R_k)
    $\mathbf{x}_k \leftarrow$ recombine_x (R_k)
    $\mathbf{s}_k \leftarrow$ mutate_s $(\mathbf{s}_k)$
    $\mathbf{x}_k \leftarrow$ mutate_x $(\mathbf{x}_k)$
    {Box-constraint handling}
    $f_k \leftarrow f(\mathbf{x}_k)$
    $P_\mathrm{o} \leftarrow P_\mathrm{o} \cup \{(\mathbf{x}_k, \mathbf{s}_k, f_k)\}$
  end for
  if comma–selection **then**
    $P_\mathrm{p} \leftarrow$ select (P_o, μ)
  else if plus–selection **then**
    $P_\mathrm{p} \leftarrow$ select $(P_\mathrm{p} \cup P_\mathrm{o}, \mu)$
  end if
  $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt}) \leftarrow$ select_best (P_p)
  $g \leftarrow g + 1$
end while
return $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt})$

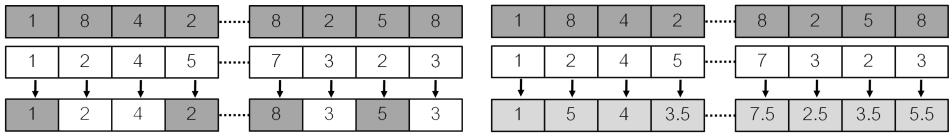

Figure 4.2: Two recombination types, illustrated for the recombination of two parents. Left: discrete recombination of two vectors where each element of the new vector is a copy of the corresponding element of one of the two parents. Right: intermediate recombination of two vectors where each element of the new vector is the average of the corresponding elements of the parents.

However, besides the object variables, also the strategy parameters are included in the mutation scheme. Moreover, as these parameters are used to control the distribution from which the perturbations are drawn, the strategy parameters are mutated before the object variables in order to achieve co-evolution of the strategy parameters. For isotropic mutations, there is only one strategy parameter, $\mathbf{s} = (\sigma)$. It scales the magnitude of the isotropic perturbations (i.e., it controls the stepsize). The combined mutation operation for an offspring with strategy parameter σ and object variables $\mathbf{x}$ is specified as:

$$\sigma = \sigma \exp\left(\tau \mathcal{N}(0,1)\right), \tag{4.4}$$

$$\mathbf{x} = \mathbf{x} + \sigma\mathbf{z}\ ,\ \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.5}$$

Here, σ is mutated using the a log-normal distribution. The parameter τ is the so-called learning parameter, which controls the magnitude of the changes of σ and is recommended to be chosen as $\tau \propto 1/\sqrt{n}$. In this work, we adopt the setting $\tau = 1/\sqrt{2n}$, which is recommended for multimodal fitness landscapes (see [BS02]).

Besides the isotropic distribution, two other variants exist: a distribution with different scalings along the main axes and a distribution with correlation of the variables. The philosophy behind using more involved distributions is that these offer the possibility to align the mutation distribution with the iso-lines of the fitness landscape, therewith speeding up the search. When denoting the mutation of the object variables as the addition of a Gaussian random variable sampled from $\sigma\mathcal{N}(\mathbf{0}, \mathbf{C})$, with $\mathbf{C}$ being the covariance matrix of the distribution, the three mutation variants can be distinguished by the form of $\mathbf{C}$. For isotropic mutations, $\mathbf{C}$ equals the identity matrix. For scaled uncorrelated mutations, $\mathbf{C}$ is a diagonal matrix. For correlated (scaled and rotated mutations), $\mathbf{C}$ is a semi-definite covariance matrix. Figure 4.3 exemplifies the differences between these three types of mutations for the mutation of a two-dimensional vector of object variables $\mathbf{x}$. It shows the density map for the generated mutations, with height lines for points of equal probability. The mutation mechanisms for the other two mutation types are described in, e.g., [Bäc96, BS02, ES03].

Regarding notation, when using mutative self-adaptation as described above, this is sometimes incorporated into the notation as $(\mu/\rho\overset{+}{,}\lambda)$-SA-Evolution Strategy. In particular, when using isotropic mutations with only one strategy parameter, σ, this is written as the $(\mu/\rho\overset{+}{,}\lambda)$-

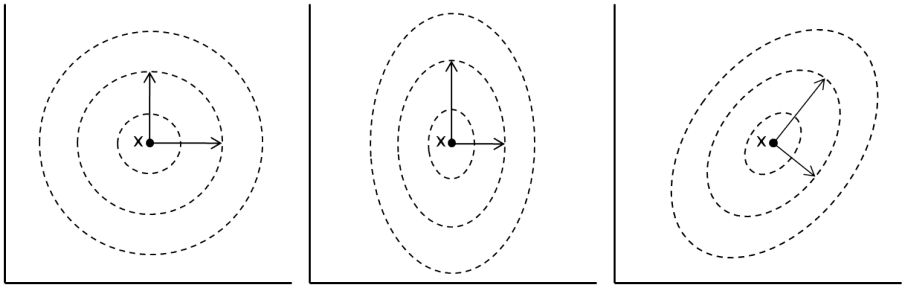

Figure 4.3: The effect of three mutation types, shown by means of iso-lines of the density map of the mutation distribution for a two-dimensional vector of object variables $\mathbf{x}$. Left: isotropic distribution. Center: scaled uncorrelated mutations. Right: correlated mutations.

σSA-Evolution Strategy.

Population size: Traditionally, the population size of Evolution Strategies is set to $\mu = 15$ and $\lambda = 100$, adhering to the recommended ratio $\mu/\lambda \approx 1/7$ (see, e.g., [Bäc96]). When using only a single stepsize parameter, smaller population sizes become also possible, such as $(1\overset{+}{,}10)$-, $(4\overset{+}{,}28)$-, and $(5\overset{+}{,}35)$-strategies.

Canonical settings: There are many different $(\mu/\rho\overset{+}{,}\lambda)$-SA-Evolution Strategy variants possible. In this work, we focus on one variant in particular, namely the $(\mu/\rho_{DI},\lambda)$-σSA-ES, with $\rho = 2$, $\mu = 5$, and $\lambda = 35$. It uses discrete recombination of the object variables and intermediate recombination of the strategy parameters, mutation is based on an isotropic multivariate Gaussian distribution, (which involves only one strategy parameter, σ), and the selection type is comma-selection. The initialization of the population is done by

$$\begin{cases} \mathbf{x}_k \sim \mathcal{U}\left(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u}\right) \\ \sigma_k = \frac{||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||}{3\sqrt{n}} \end{cases}, \; k = 1, \ldots, \mu. \tag{4.6}$$

4.2.3 The Covariance Matrix Adaptation Evolution Strategy

The *Covariance Matrix Adaptation* Evolution Strategy (CMA-ES), proposed by Hansen and Ostermeier [HO96, HO01], can be seen as a second generation Evolution Strategy. It is a $(\mu/\mu_\mathrm{W},\lambda)$ Evolution Strategy that uses comma-selection and global weighted intermediate recombination (indicated by the subscript W) in which all offspring are generated from the same recombinant $\langle\mathbf{x}\rangle_\mathrm{W}$, computed as the weighted center of mass of the μ selected individuals, i.e.,

$$\langle\mathbf{x}\rangle_\mathrm{W} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}, \tag{4.7}$$

with $\sum_{i=1}^{\mu} w_i = 1$ and $\mathbf{x}_{i:\lambda}$ denoting the object variables of the ith best individual. Within the CMA-ES, the offspring are mutated copies of this recombinant, generated as

$$\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.8}$$

$$\mathbf{y}_k = \mathbf{C}^{\frac{1}{2}}\mathbf{z}, \tag{4.9}$$

$$\mathbf{x}_k = \langle \mathbf{x} \rangle_{\mathrm{W}} + \sigma \mathbf{y}_k, \tag{4.10}$$

for $k = 1, \ldots, \lambda$. Hence, the mutations are drawn from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{C})$. The motivation for using such a mutation mechanism is to adapt the mutation distribution to the local curvature of the fitness landscape. When this is achieved, this has as effect that the mutations are taken along the gradient direction, which significantly increases the convergence speed of the algorithm, especially for ill-conditioned problems. In order to use such a mutation scheme, a proper adaptation scheme of the covariance matrix $\mathbf{C}$ and of the stepsize σ is required. In the CMA-ES, these two issues are handled by two different control mechanisms. For the former, a *covariance matrix adaptation* (CMA) scheme is used, for the latter, a stepsize adaptation mechanism named *cumulated stepsize adaptation* (CSA) is used.

The covariance matrix determines the direction of the mutations. It is initialized with $\mathbf{C} = \mathbf{I}$ and updated each generation based on a weighted empirical covariance estimation of the μ best individuals of the population (*rank-μ-update*) and on the direction of the so-called evolution path $\mathbf{p}_{\mathrm{c}}$ (*rank-one-update*). The evolution path (initialized with $\mathbf{p}_{\mathrm{c}} = \mathbf{0}$) is loosely defined as the sequence of successive steps taken by the population over a number of generations. It is a vector that tracks the direction followed by the population in the past few generations. The update of the evolution path and the covariance matrix are done in a cumulative way, with

$$\mathbf{p}_{\mathrm{c}} = (1 - c_{\mathrm{c}})\mathbf{p}_{\mathrm{c}} + \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})\mu_{\mathrm{eff}}}\langle \mathbf{y} \rangle_{\mathrm{W}}, \tag{4.11}$$

$$\mathbf{C} = (1 - c_1 - c_\mu)\mathbf{C} + c_1 \mathbf{p}_{\mathrm{c}}\mathbf{p}_{\mathrm{c}}^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^{\mathrm{T}}. \tag{4.12}$$

Here, $\langle \mathbf{y} \rangle_{\mathrm{W}} = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$ represents the direction of the step taken by the population's center of mass $\langle \mathbf{x} \rangle_{\mathrm{W}}$. The parameter μ_{eff} is the so-called *variance effective selection mass*. The parameters c_{c}, c_1, and c_μ are the cumulation factors for the evolution path update, the rank-one-update, and the rank-μ-update respectively.

The stepsize parameter σ scales the mutations. The cumulated stepsize adaptation mechanism also uses the so-called conjugate evolution path $\mathbf{p}_\sigma$ (initialized with $\mathbf{p}_\sigma = \mathbf{0}$), which registers both the length and direction of the evolution path. When the length of the evolution path $\mathbf{p}_\sigma$ is short, then steps are taken in opposite directions that cancel each other out, yielding an ineffective circling behavior. In this case, the stepsize is decreased. When the evolution path length is large, consecutive steps are taken in the same direction, from which it can be concluded that the stepsize can be increased for faster convergence. As a reference for the evolution path length, the CSA mechanism considers the path length that would occur under

Technical Note 4.1: Parameter settings of the CMA-ES

Default population size:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor,\ \mu = \lfloor \mu' \rfloor,\ \mu' = \frac{\lambda}{2}. \tag{4.15}$$

Recombination weights:

$$w_i = \frac{w_i}{\sum_{j=1}^{\mu} w_j'},\ w_j' = \ln(\mu' + 0.5) - \ln j,\ \text{for } i = 1, \ldots, \mu. \tag{4.16}$$

Variance effective selection mass:

$$\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}. \tag{4.17}$$

Covariance matrix adaptation parameters:

$$c_{\text{c}} = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n},\ c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}},\ c_\mu = \min\left(1 - c_1, 2\frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \mu_{\text{eff}}}\right). \tag{4.18}$$

Stepsize adaptation parameters:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5},\ d_\sigma = 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma. \tag{4.19}$$

random selection. The CSA update mechanism can be summarized as

$$\mathbf{p}_\sigma = (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{-\frac{1}{2}}\langle \mathbf{y} \rangle_{\text{W}}, \tag{4.13}$$

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma||}{\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]} - 1\right)\right). \tag{4.14}$$

The update of the evolution path length $\mathbf{p}_\sigma$ is similar to the update of $\mathbf{p}_{\text{c}}$. The stepsize update uses an exponential update, with d_σ being a damping factor, and $\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]$ being the expected length of a random vector drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

The setting of the parameters used within the CMA-ES are given in Technical Note 4.1. For more details regarding these settings, the reader is referred to [HO96, HO01]. Finally, for the sake of completeness, Algorithm 4.3 provides an algorithmic description of the CMA-ES.

4.2.4 Box-Constraint Handling

For real-parameter optimization problems, the search space is bounded by the hyperbox $[\mathbf{x}_{\text{l}}, \mathbf{x}_{\text{u}}]$ and in many scenarios it is desirable to only generate candidate solutions that lie within this hyperbox. Since mutation can yield solutions that are not within this hyperbox,

Algorithm 4.3: The CMA Evolution Strategy

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, lower bounds $\mathbf{x}_\mathrm{l} \in \mathbb{R}^n$, upper bounds $\mathbf{x}_\mathrm{u} \in \mathbb{R}^n$

Output: best solution found $\mathbf{x}_\mathrm{opt}$ with objective function value f_opt

Set parameters: the parameters λ, μ, μ', $w_1, \ldots, w_\mu, \mu_\mathrm{eff}$, c_c, c_1, c_μ, c_σ, d_σ are set according to Technical Note 4.1

Initialize: $g \leftarrow 0$, $\mathbf{p}_\mathrm{c} = \mathbf{0}$, $\mathbf{C} = \mathbf{I}$, $\mathbf{p}_\sigma = \mathbf{0}$, $\sigma = ||\mathbf{x}_\mathrm{u} - \mathbf{x}_\mathrm{l}||/(3\sqrt{n})$, $\langle \mathbf{x} \rangle_\mathrm{W} \sim \mathcal{U}(\mathbf{x}_\mathrm{l}, \mathbf{x}_\mathrm{u})$

$g \leftarrow 0$

while not terminate **do**

  for $k = 1 \to \lambda$ **do**

    $\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

    $\mathbf{y}_k \leftarrow \mathbf{C}^{\frac{1}{2}} \mathbf{z}_k$

    $\mathbf{x}_k \leftarrow \langle \mathbf{x} \rangle_\mathrm{W} + \sigma \mathbf{y}_k$

    $\{$Box-constraint handling$\}$

    $f_k \leftarrow f(\mathbf{x}_k)$

  end for

  $\langle \mathbf{y} \rangle_\mathrm{W} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$

  $\langle \mathbf{x} \rangle_\mathrm{W} \leftarrow \langle \mathbf{x} \rangle_\mathrm{W} + \sigma \langle \mathbf{y} \rangle_\mathrm{W}$

  $\{$Box-constraint handling$\}$

  $\mathbf{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c}) \mathbf{p}_\mathrm{c} + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c}) \mu_\mathrm{eff}} \langle \mathbf{y} \rangle_\mathrm{W}$

  $\mathbf{C} \leftarrow (1 - c_1 - c_\mu) \mathbf{C} + c_1 \mathbf{p}_\mathrm{c} \mathbf{p}_\mathrm{c}^\mathrm{T} + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^\mathrm{T}$

  $\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma) \mu_\mathrm{eff}} \mathbf{C}^{-\frac{1}{2}} \langle \mathbf{y} \rangle_\mathrm{W}$

  $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{||\mathbf{p}_\sigma||}{\mathbf{E}[||\mathcal{N}(\mathbf{0}, \mathbf{I})||]} - 1 \right) \right)$

  $g \leftarrow g + 1$

  $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt}) \leftarrow (\mathbf{x}_{1:\lambda}, f_{1:\lambda})$

end while

return $(\mathbf{x}_\mathrm{opt}, f_\mathrm{opt})$

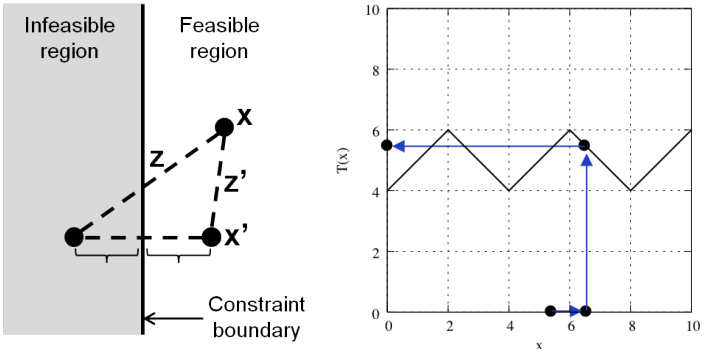

Figure 4.4: Left: visualization of reflection box-constraint handling in a two-dimensional search space. Right: an illustration of the working mechanism of the transformation function $\mathrm{T}_{[a,b]}$ with $a = 4$ and $b = 6$ (right figure, courtesy of Li [Li09]).

an additional *box-constraint handling* mechanism is needed in order to accomplish this.

For box-constraint handling, two straightforward methods are to reject mutations that fall outside the box or to apply a cutoff rule forcing mutated offspring back to the nearest point on the constraint boundary. In this work we use a *reflection* mechanism (see [Li09]) in which mutations are mirrored in the constraint boundaries. That is, the ith element of a candidate solution $\mathbf{x}$ is transformed as:

$$(\mathbf{x})_i = \mathrm{T}_{[(\mathbf{x}_\mathrm{l})_i,(\mathbf{x}_\mathrm{u})_i]}\left((\mathbf{x})_i\right), \tag{4.20}$$

with

$$\mathrm{T}_{[a,b]}(x) = x + y' \cdot (b-a), \tag{4.21}$$

$$y' = \begin{cases} |y - \lfloor y \rfloor| & \text{if } \lfloor y \rfloor \bmod 2 = 0 \\ 1 - |y - \lfloor y \rfloor| & \text{otherwise} \end{cases}, \; y = \frac{x-a}{b-a}. \tag{4.22}$$

Figure 4.4 visualizes the working mechanism of reflection.

The reason for choosing this mechanism is that it is relatively simple to implement and it preserves much of the normal dynamics of Evolution Strategies, also near the constraint boundaries. Moreover, also in practical scenarios where the optimizer might be located very close to a box-constraint boundary, this method is well applicable.

When using this box-constraint handling mechanism within the CMA-ES, note that the recombinant $\langle \mathbf{x} \rangle_\mathrm{W}$ should be computed by addition of $\sigma \langle \mathbf{y} \rangle_\mathrm{W}$, not as the weighted mean of the selected offspring, and handled separately (also with reflection). For the $(\mu/\rho^+,\lambda)$-SA-Evolution Strategy, reflection only needs to be applied to the offspring.

4.3 Summary and Discussion

This chapter has provided a brief introduction to Evolutionary Algorithms and in particular Evolution Strategies, which are designed to solve single-objective real-parameter optimization problems. It has introduced the two main algorithmic schemes that will be considered throughout this work in the context of robust optimization; the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES.

The $(5/2_{DI}, 35)$-σSA-ES is an element of the class of classical Evolution Strategies as proposed by Schwefel [Sch77]. It is a population based Evolution Strategy in which the strategy parameters are included in the encoding of the individuals in order to co-evolve them together with the object variables (self-adaptation). It uses two-parent recombination with discrete recombination of the object variables and intermediate recombination of the strategy parameters. For mutation it uses an isotropic multivariate Gaussian distribution scaled according to the stepsize parameter.

The CMA-ES [HO96, HO01] can be seen as a derandomized version a classical Evolution Strategy. The main difference is that it adopts a different way of controlling the strategy parameters. For the mutation, it uses a multivariate Gaussian distribution based on a full covariance matrix and scaled with a stepsize parameter. The covariance matrix is adapted based on the direction of successful mutations (rank-one and rank-μ update) and the stepsize parameter is adapted based on the length of the evolution path (cumulative stepsize adaptation). Furthermore, it adopts global weighted intermediate recombination for the object variables and uses a small population size as compared to the $(5/2_{DI}, 35)$-σSA-ES.

The two algorithmic schemes are considered to be instantiated canonically as described in Section 4.2.2 and Section 4.2.3, respectively, and are used in combination with reflection box-constraint handling as discussed in Section 4.2.4.

In this work, we study to what extent the $(5/2_{DI}, 35)$-σSA-ES and the CMA-ES are suitable in robust optimization scenarios, or how they should be adapted in order to make them such. Two scenarios are considered in particular: optimization of noisy objective functions and finding robust optima.