Evol. Intel.
DOI 10.1007/s12065-010-0035-y

REVIEW ARTICLE

Evolutionary self-adaptation: a survey of operators and strategy parameters

Oliver Kramer

Received: 17 March 2009 / Revised: 16 October 2009 / Accepted: 18 December 2009

Abstract The success of evolutionary search depends on adequate parameter settings. Ill conditioned strategy parameters decrease the success probabilities of genetic operators. Proper settings may change during the optimization process. The question arises if adequate settings can be found automatically during the optimization process. Evolution strategies gave an answer to the online parameter control problem decades ago: self-adaptation. Self-adaptation is the implicit search in the space of strategy parameters. The self-adaptive control of mutation strengths in evolution strategies turned out to be exceptionally successful. Nevertheless, for years self-adaptation has not achieved the attention it deserves. This paper is a survey of self-adaptive parameter control in evolutionary computation. It classifies self-adaptation in the taxonomy of parameter setting techniques, gives an overview of automatic online-controllable evolutionary operators and provides a coherent view on search techniques in the space of strategy parameters. Beyer and Sendhoff's covariance matrix self-adaptation evolution strategy is reviewed as a successful example for self-adaptation and exemplarily tested for various concepts that are discussed.

Keywords Self-adaptation · Parameter control · Mutation · Crossover · Evolution strategies · Covariance matrix self-adaptation

O. Kramer (✉)
Technische Universität Dortmund, Dortmund, Germany
e-mail: oliver.kramer@tu-dortmund.de

1 Introduction

Self-adaptation is a famous concept—not only known in computer science—meaning that a system is capable of adapting itself autonomously. In evolutionary optimization the term has a defined meaning: the evolutionary online-control of so called *strategy variables*. Strategy variables define properties of an evolutionary algorithm, e.g., of its genetic operators,[1] of the selection operator or populations sizes. Their online control during the search can improve the optimization process significantly. Self-adaptation was originally introduced by Rechenberg [60] and Schwefel [68] for evolution strategies (ES), later by Fogel [25] for evolutionary programming (EP). The goal of this paper is to give a deeper insight into the underlying principles by overviewing typical self-adaptive operators and techniques for the search in the space of strategy parameters. Starting from a definition of self-adaptation and an introduction to the covariance matrix self-adaptation evolution strategy (CMSA-ES) by Beyer and Sendhoff [16] as a successful example for self-adaptation in real-valued solution spaces, this paper is structured as follows. Section 2 classifies self-adaptation within the taxonomy of parameter setting methodologies by Eiben et al. [22]. Section 3 gives a survey of strategy parameters that are typically controlled by means of self-adaptation, from mutation strengths in real-valued domains to the self-adaptive aggregation mechanisms for global parameters. Section 4 concentrates on techniques to discover the space of strategy variables and to exploit knowledge from the search. Section 5 focuses on premature convergence and describes situations when self-adaptation might fail.

[1] Typically, genetic operators are crossover/recombination, mutation, inversion, gene deletion and duplication.

1.1 What is self-adaptation?

Basis of evolutionary search is a population $\mathcal{P}$ of μ individuals $\mathbf{a}_1$ ..., $\mathbf{a}_\mu$.[2] Each $\mathbf{a}$ consists of an N-dimensional objective variable vector $\mathbf{x}$ that defines a solution for the optimization problem of dimension N. The idea of self-adaptation is the following: a vector $\boldsymbol{\sigma}$ of strategy variables[3] is bound to each chromosome [22]. Both, objective and strategy variables, undergo variation and selection as individual $\mathbf{a} = (\mathbf{x}, \boldsymbol{\sigma})$. Individuals inherit their whole genetic material, i.e., the chromosome $\mathbf{x}$ defining the solution, *and* the strategy variable vector $\boldsymbol{\sigma}$. Self-adaptation is based on the following principle: good solutions more likely result from good than from bad strategy variable values. Bound to the objective variables, these good parameterizations have a high probability of being selected and inherited to the following generation. Self-adaptation becomes an implicit evolutionary search for optimal strategy variable values. These values define properties of the evolutionary algorithm, e.g., mutation strengths, or global parameters like selection pressure or population sizes. In Sect. 3.4 we will see examples of successful self-adaptive control of global parameters. Self-adaptive strategy variables are also known as *endogenous*, i.e., evolvable, in contrast to *exogenous* parameters, which are kept constant during the optimization run [15]. A necessary condition for successful strategy variable exploration is birth surplus, i.e., the number of offspring solutions is higher than the number of parents. In ES the relation is usually $\mu/\lambda \approx 1/4$ to 1/7. For the CMSA-ES that we will get to know in detail later in this section, Beyer et al. [16] recommend $\mu/\lambda = 1/4$. A birth surplus increases the chance that both objective and strategy variables are explored effectively.

1.2 A short history of self-adaptation

The development of parameter adaptation mechanisms began in 1967, when Reed, Toombs and Baricelli [62] learned to play poker with an EA. The genome contained strategy parameters determining probabilities for mutation and crossover with other strategies. In the same year, Rosenberg [64] proposed to adapt the probability for applying crossover. With the 1/5-th rule Rechenberg [60] introduced an adaptation mechanism for step size control of ES, see Sect. 2.2.2. Weinberg [76] as well as Mercer and Sampson [50] were the first who introduced meta-evolutionary approaches. In meta-evolutionary methods an outer EA controls the parameters of an inner one that optimizes the original problem.

Today, the term self-adaptation is commonly associated with the self-adaptation of mutative step sizes for ES like introduced by Schwefel [68] in 1974. For numerical representations the self-adaptation of mutation parameters seems to be an omnipresent feature. After ES, Fogel introduced self-adaptation to EP [25]. However, for binary-coded EAs self-adaptation has not grown to a standard method. Nevertheless, there are several approaches to incorporate it into binary representations, e.g., by Bäck [8, 9], Smith and Fogarty [73], Smith [71] and Stone and Smith [75]. Schaffer and Morishima [67] control the number and location of crossover points self-adaptively in their approach called *punctuated crossover*. Ostermeier et al. [58] introduced the cumulative path-length control, an approach to derandomize the adaptation of strategy parameters. Two algorithmic variants were the results of their attempts: the cumulative step-size adaptation (CSA) [26] and later the covariance matrix adaptation evolution strategy (CMA-ES) [57]. Beyer et al. [16] introduced the self-adaptive variant CMSA-ES, that will be presented in the next section as an example for a successful self-adaptive evolutionary method. Many successful examples of CMA-ES applications can be reported, e.g., in flow optimization [41] or in optimization of kernel parameters for classification problems [51].

1.3 CMSA-ES—an example of self-adaptive parameter control

This section introduces a successful self-adaptive evolutionary algorithm for real-valued representations, the CMSA-ES that has recently been proposed by Beyer et al. [16]. It stands exemplarily for self-adaptation, but furthermore allows us to evaluate and discuss self-adaptive concepts reviewed in later parts of this paper. The CMSA-ES is a variant of the CMA-ES by Hansen and Ostermeier [57]. Covariance matrix adaptation techniques are based on the covariance matrix computation of the differences of the best solutions and the parental solution. Although the CMA-ES turned out to be the state-of-the-art evolutionary optimization algorithm for many years, the original algorithm suffers from parameters with little theoretical guidance [16]. The CMSA-ES is based on a self-adaptive step size control of these parameters.

Figure 1 shows the pseudocode of the CMSA-ES by Beyer et al. [16]. After initialization, λ candidate solutions are produced: First, with the help of the global[4] self-adaptive step size $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ each individual $\mathbf{a}_i$ is

[2] According to a tradition in ES, μ is the size of the parental population, while λ is the size of the offspring population.

[3] Originally, σ denotes step sizes in ES. We use this symbol for all kinds of strategy variables in this paper.

[4] The concept of global multi-recombination will be explained in Sect. 4.2.

```
Start
  Initialize
  Repeat
    For $i = 1$ To $\lambda$
      $\sigma_i = \hat{\sigma} \cdot e^{\tau \mathcal{N}_i(0,1)}$
      $\mathbf{s}_i = \sqrt{\mathbf{C}}\mathcal{N}_i(\mathbf{0},\mathbf{1})$
      $\mathbf{z}_i = \sigma_i \mathbf{s}_i$
      $\mathbf{y}_i = \mathbf{y} + \mathbf{z}_i$
      $f_i = f(\mathbf{y}_i)$
    Next
    Sort population with regard to fitness $f$
    $\hat{\mathbf{z}} = \frac{1}{\mu}\sum_{j=1}^{\mu} \mathbf{z}_{j:\lambda}$
    $\hat{\sigma} = \frac{1}{\mu}\sum_{j=1}^{\mu} \sigma_{j:\lambda}$
    $\mathbf{S} = \frac{1}{\mu}\sum_{j=1}^{\mu} \mathbf{s}_{j:\lambda}\mathbf{s}_{j:\lambda}^T$
    $\mathbf{y} = \mathbf{y} + \hat{\mathbf{z}}$;
    $\mathbf{C} = (1 - \frac{1}{\tau_c})\mathbf{C} + \frac{1}{\tau_c}\mathbf{S}$
  Until termination condition
End
```

Fig. 1 Pseudocode of the CMSA-ES by Beyer and Sendhoff [16]

combined and mutated with a log-normally[5] distributed step size σ_i:

$$\sigma_i = \hat{\sigma} \cdot e^{\tau \mathcal{N}_i(0,1)}, \tag{1}$$

depending on a global step size $\hat{\sigma} = \frac{1}{\mu}\sum_{j=1}^{\mu} \sigma_{j:\lambda}$ that is the arithmetic mean of the step sizes from the μ best of λ offspring solutions[6] of the previous generation. Then, correlated random directions $\mathbf{s}_i$ are generated with the help of the covariance matrix $\mathbf{C}$ by multiplication of the Cholesky decomposition $\sqrt{\mathbf{C}}$ with the standard normal vector $\mathcal{N}_i(\mathbf{0},\mathbf{1})$:

$$\mathbf{s}_i = \sqrt{\mathbf{C}}\mathcal{N}_i(\mathbf{0},\mathbf{1}). \tag{2}$$

This random direction is scaled in length with regard to the self-adaptive step size σ_i:

$$\mathbf{z}_i = \sigma_i \mathbf{s}_i. \tag{3}$$

The resulting vector $\mathbf{z}_i$ is added to the global parent $\mathbf{y}$:

$$\mathbf{y}_i = \mathbf{y} + \mathbf{z}_i. \tag{4}$$

Finally, quality f_i of solution $\mathbf{y}_i$ is evaluated. When λ offspring solutions have been generated, the μ best solutions are selected and their components $\mathbf{z}_i$ and σ_i are recombined. Beyer and Sendhoff propose to apply global recombination, i.e., the arithmetic mean of each parameter is calculated. The outer product $\mathbf{s}\mathbf{s}^T$ of the search directions is computed for each of the μ best solutions and the matrices are averaged afterwards:

$$\mathbf{S} = \frac{1}{\mu}\sum_{j=1}^{\mu} \mathbf{s}_{j:\lambda}\mathbf{s}_{j:\lambda}^T. \tag{5}$$

Important is the final covariance matrix update step:

$$\mathbf{C} = \left(1 - \frac{1}{\tau_c}\right)\mathbf{C} + \frac{1}{\tau_c}\mathbf{S}. \tag{6}$$

Parameter τ_c balances between the last covariance matrix $\mathbf{C}$ and the outer product of the search direction of the μ best solutions. These steps are iterated until a termination condition is satisfied. The CMSA-ES combines the self-adaptive step size control with a simultaneous update of the covariance matrix. To show the benefits of a self-adaptive approach we will compare the CMSA-ES to an ES with constant step sizes in Sect. 4. Initially, the covariance matrix $\mathbf{C}$ is chosen as the identity matrix $\mathbf{C} = \mathbf{I}$. The learning parameter τ defines the mutation strength of the step sizes σ_i. Beyer and Sendhoff recommend:

$$\tau_c = \frac{N(N+1)}{2\mu}. \tag{7}$$

For the *Sphere* problem[7] the optimal learning parameter is $\tau = \frac{1}{\sqrt{2N}}$ [15]. Smaller values increase the approximation speed at the beginning of the search. The learning parameter τ_c influences the covariance matrix. In Sect. 3.4 we will control the global parameter τ_c self-adaptively using an aggregation proposed by Eiben et al. [21].

1.4 Theory of self-adaptation

Only few theoretical investigations of self-adaptation exist. Most of them concern continuous search domains and the analysis of mutation strengths in ES. As Beyer and Schwefel [15] state, the analysis of evolutionary algorithms including the mutation control part is a difficult task. In the following, we give an overview of the main theoretical results. Some results about premature convergence presented in the last section, e.g., by Rudolph [65], also belongs to the line of theoretical research.

1.4.1 Dynamical systems

Beyer [12] gives a detailed analysis of the (1, λ)-ES-self-adaptation on the *Sphere* problem in terms of progress rate and self-adaptation response, also considering the dynamics with fluctuations. The analysis reveals various characteristics of the SA-(1, λ) $-$ ES, e.g., it confirms that the choice for the learning parameter $\tau = c/\sqrt{N}$ is reasonable. Beyer and Meyer-Nieberg [14] present results for σ-self-adaptation on the sharp ridge problem for a (1, λ)-ES without recombination. They use *evolution equation*

[5] For log-normal mutation of step sizes, see Sect. 4.1.2.

[6] The index j denotes the index of the j-th ranked individual of the λ offspring individuals with regard to fitness $f(\mathbf{x}_j)$.

[7] The *Sphere* problem is the continuous optimization problem to find an $\mathbf{x} \in \mathbb{R}^N$ minimizing $f(\mathbf{x}) := \sum_{i=1}^{N} x_i^2$.

approximations and show that the radial and the axial progress rate as well as the self-adaptation response depend on the distance to the ridge axis. A further analysis reveals that the ES reaches a stationary normalized mutation strength. According to their analysis the problem parameter d of the ridge function determines whether the ES concentrates on decreasing the distance to the ridge axis or increases the speed along the ridge axis. If d is smaller than a limit depending on the population size, the stationary step size leads to an increase of distance d to the ridge. Beyer and Deb [13] derive evolution equations as well as population mean and variance of a number of evolutionary operators, e.g., for real-parameter crossover operators. They also recommend appropriate strategy parameter values based on the comparison of various self-adaptive evolutionary algorithms.

1.4.2 Runtime analysis

Rechenberg [60] proved that the optimal step size of an ES depends on the distance to the optimum and proposed the 1/5-th adaptation rule. Jägersküpper [37] proved that the (1+1)-ES using Gaussian mutations adapted by the 1/5-rule needs a linear number of steps, i.e., $O(n)$, to halve the approximation error on the *Sphere* problem. The bound $O(n)$ holds with "overwhelming probability of $1 - e^{-\Omega(n^{1/3})}$". Later, Jägersküpper [38] proved the bound $O(n{\cdot}\lambda\sqrt{\ln\lambda})$ for the $(1+\lambda)$-ES with Rechernberg's rule. He proposes a modified 1/5-th rule for λ descendants as the regular algorithm fails for more than a single isotropic mutation. A further runtime analysis is reported by Auger [3] who investigated the $(1,\lambda)$-SA-ES on the one-dimensional *Sphere* problem. She proved sufficient conditions on the algorithm's parameters, and showed that the convergence is $\frac{1}{t}\ln(\|X_t\|)$ with parent X_t at generation t. Her proof technique is based on Markov chains for continuous search spaces and the Foster-Lyapunov drift conditions that allow stability property proofs for Markov chains. The main result is the log-linear type of the convergence and an estimation of the convergence rate. Although the proof is quite sophisticated, it does not show how the number of steps scales depending on the search space dimensions.

The work of Semenov and Terkel [70] is based on a stochastic Lyapunov function. A supermartingale is used to investigate the convergence of a simple self-adaptive evolutionary optimization algorithm. A central result of their research is that the convergence velocity of the analyzed self-adaptive evolutionary optimization algorithm is asymptotically exponential. They use Monte-Carlo simulations to validate the confidence of the martingale inequalities numerically, because they cannot be proven.

2 Self-adaptation in the taxonomy of parameter settings

Carefully formulated adaptive methods for parameter control can accelerate the search significantly, but require knowledge about the problem—that might often not be available. Many black-box optimization problems are real black boxes and no knowledge about the solution space is available. It would be comfortable, if the algorithms tuned their parameters autonomously. In this section we overview the most important parameter setting techniques in the field of evolutionary algorithms. From this broader point of view we are able to integrate self-adaptation into a taxonomy of parameter setting methods. Getting to know the related tuning and control strategies helps to understand the importance of self-adaptive parameter control. A more detailed overview of parameter setting gives De Jong [20] overviewing 30 years of research in this area. Also Eiben et al. [23] give a comprehensive overview of parameter control techniques. Bäck [4] gave an overview of self-adaptation over 10 years ago.

Eiben et al. [22] asserted two main types of parameter settings techniques. Their taxonomy is based on an early taxonomy of Angeline [1]. Evolutionary algorithms are specified by various parameters that have to be *tuned before* or *controlled during* the run of the optimization algorithm. Figure 2 shows a classical taxonomy of parameter setting techniques complemented on the parameter tuning branch.

2.1 Parameter tuning

Many parameters of evolutionary heuristics are static, i.e., once defined they are not changed during the optimization process. Typical examples for static parameters are population sizes or initial strategy parameter values. The disadvantage of setting the parameters statically is the lack of flexibility *during* the search process. This motivates automatic control strategies like self-adaptation.

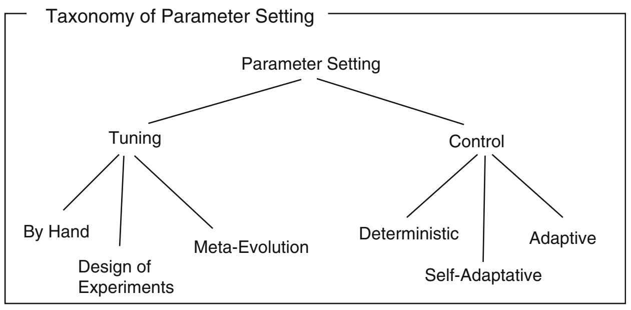

Fig. 2 Taxonomy of parameter setting methods for evolutionary algorithms by Eiben et al. [22] complemented on the parameter tuning branch

2.1.1 Tuning by hand

In many cases parameters can be tuned by hand. In this case, the influence on the behavior of the algorithm depends on human experience. But user defined settings might not be the optimal ones. Mutation operators are frequently subject to parameter tuning. At the beginning of the history of evolutionary computation, researchers argued about proper mutation rates. De Jong's [40] recommendation was the mutation strength $p_m = 0.001$, Schaffer et al. [66] recommended $0.005 \leq p_m \leq 0.01$, and Grefenstette [28] $p_m = 0.01$. Mühlenbein [54] suggested to set the mutation probability $p_m = 1/l$ depending on the length l of the representation. The idea appeared to control the mutation rate during the optimization run as the optimal rate might change during the optimization process or different rates are reasonable for different problems.

2.1.2 Design of experiments

Today, statistical tools like design of experiments (DoE) support the parameter tuning process. Bartz-Beielstein [5] gives a comprehensive introduction to experimental research in evolutionary computation. An experimental design is the layout of a detailed experimental plan in advance of doing the experiment. DoE starts with the determination of objectives of an experiment and the selection of parameters (factors) for the study. The quality of the experiment (response) guides the search to find appropriate settings. In an experiment, DoE deliberately changes one or more factors, in order to observe the effect the changes have on one or more *response* variables. The response can be defined as the quality of the results, e.g., average fitness values at a given generation or convergence ratios. Well chosen experimental designs maximize the amount of *information* that can be obtained for a given amount of experimental effort. Bartz-Beielstein et al. [6, 7] developed a parameter tuning method for stochastically disturbed algorithm output, the Sequential Parameter Optimization (SPO). SPO has successfully been applied in many applications. Preuss et al. [59] use SPO to tune self-adaptation for a binary coded evolutionary algorithm. It combines classical regression methods and statistical approaches.

2.1.3 Meta-evolution

Meta-evolutionary algorithms, also known as nested evolutionary algorithms, belong to the tuning branch of our taxonomy. In meta-evolutionary algorithms the evolutionary optimization process takes place on two levels [61]: an outer optimization algorithm tunes the parameters of an embedded algorithm. Nested approaches are able to successfully tune the parameters of the inner optimizer, but are quite inefficient. The *isolation time* defines how long the embedded algorithm is allowed to optimize the objective function. Its response guides the parameter search of the outer parameter optimization algorithm. An early meta-GA approach stems from Grefenstette [28], who optimized the parameters of a classical genetic algorithm that itself solved a suite of problems. Coello [17] makes use of nested ES for adapting the factors of a penalty function for constrained problems. Herdy [32] analyzed the behavior of meta-evolution strategies on a test function set and analyzed the meta-(1, λ)-ES on the *Sphere* problem using results from the theory of progress rates. Nannen and Eiben [19, 55, 56] proposed the relevance estimation and value calibration (REVAC) method to estimate the sensitivity of parameters and the choice of their values. It is based on information theory, i.e., REVAC estimates the expected performance when parameter values are chosen from a probability density distribution $\mathcal{C}$ with maximized Shannon entropy. Hence, REVAC is an estimation of distribution algorithm. It iteratively refines a joint distribution $\mathcal{C}$ over possible parameter vectors beginning with a uniform distribution and giving an increasing probability to enlarge the expected performance of the underlying evolutionary algorithm. From a vertical perspective new distributions for each parameter are built on estimates of the response surface, i.e., the fitness landscape. From a horizontal point of view in the first step the parameter vectors are evaluated according to the optimization algorithms's performance, in the second step new parameter vectors are generated with superior response.

2.2 Online parameter control

The change of evolutionary parameters *during* the run is called *online parameter control*, and is reasonable, if the conditions of the fitness landscape change during the optimization process.

2.2.1 Deterministic parameter control

Deterministic parameter control means that the parameters are adjusted according to a fixed time scheme, explicitly depending on the number of generations t. It may be useful to reduce the mutation strengths during the evolutionary search, in order to allow convergence of the population. Fogarty [24] proposed an exponentially decreasing mutation rate, i.e., $p_m(t) = \frac{1}{240} + \frac{11.375}{t^2}$. Of course, this approach is not flexible enough to tackle a broad range of problems. Hesser and Männer [33, 34] propose an adaptive scheme that depends on the length of the representation l and the population size μ:

$$p_m(t) = \sqrt{\frac{c_1 - c_3 t/2}{c_2 \quad \mu\sqrt{l}}}. \tag{8}$$

Of course, the constants c_1, c_2, c_3 have to be chosen problem-dependent, which is the obvious drawback of the approach. Another deterministic scheme for the control of mutation rates has been proposed by Bäck and Schütz [10]:

$$p_m(t) = \left(2 + \frac{l-2}{T-1} \cdot t\right), \tag{9}$$

with the overall number of generations T. Successful results have been reported for this update scheme.

In constrained solution spaces dynamic penalty functions are frequently applied. Penalties decrease the fitness of infeasible solutions. Dynamic penalties are increased depending on the number of generations, in order to avoid infeasible solutions in later phases of the search. Joines and Houck [39] propose the following dynamic penalty function:

$$\tilde{f}(\mathbf{x}) := f(\mathbf{x}) + (C \cdot t)^{\alpha} \cdot G(\mathbf{x}) \tag{10}$$

with generation t and fitness $f(\mathbf{x})$ of individual $\mathbf{x}$, the constraint violation measure $G(\mathbf{x})$ and parameters C and α. The penalty is guided by the number of generations. Although this approach may work well in many cases—as many optimization processes are quite robust to a certain parameter interval—the search can be accelerated with advanced methods. There are many other examples where parameters are controlled deterministically. Efrén Mezura-Montes et al. [53] make use of deterministic parameter control for differential evolution based constraint handling.

2.2.2 *Adaptive parameter control*

Adaptive parameter control methods make use of rules the practitioner defines for the heuristic. A feedback from the search determines magnitude and direction of the parameter change. An example for an adaptive control of endogenous strategy parameters is the 1/5-th success rule for the mutation strengths by Rechenberg [60]. Running a simple (1+1)-ES with isotropic Gaussian mutations and constant mutation steps σ the optimization process will become very slow after a few generations (also see Fig. 3 in Sect. 3.1.1). The below list shows the pseudocode of Rechenberg's adaptation rule.

```
1. Perform the (1+1)-ES for a number g of generations:
   – Keep $\sigma$ constant during $g$ generations,
   – Count number $s$ of successful mutations during this period
2. Estimate success rate $p_s$ by
   $p_s := s/g$                                        (11)
3. Change $\sigma$ according to ($0 < \vartheta < 1$):
   $\sigma := \begin{cases} \sigma/\vartheta, & \text{if } p_s > 1/5 \\ \sigma \cdot \vartheta, & \text{if } p_s < 1/5 \\ \sigma, & \text{if } p_s = 1/5 \end{cases}$     (12)
4. Go to step 1.
```

Rechenberg's rule adapts the mutation strengths in the following way: the whole population makes use of a global mutation strength σ for all individuals. If the ratio of successful candidate solutions is greater than 1/5-th, the step size should be increased, because bigger steps towards the optimum are possible, while small steps would be a waste of time. If the success ratio is less than 1/5-th the step size should be decreased. This rule is applied every g generations. It must hold $0 < \vartheta < 1$. The goal of Rechenberg's approach is to stay in the so called *evolution window* guaranteeing nearly optimal progress.[8] The optimal value for the factor ϑ depends on several conditions such as fitness landscape, number of dimensions N or number of generations g (Pseudocode of the 1/5-th rule of Rechenberg. The step sizes are increased, if the success ratio is higher than 1/5-th to allow faster progress and decreased, if the success rate is lower than 1/5-th).

2.2.3 *Self-adaptation*

If no knowledge about optimal parameter settings is available, evolutionary search in the space of strategy variables is a reasonable undertaking. In the taxonomy of parameter setting techniques self-adaptation plays the role of the stochastic online control of strategy parameters towards a parameter-free evolutionary optimization meta-heuristic.

3 Self-adaptive strategy parameters

This section gives a survey of strategy parameters that are typically controlled by means of self-adaptation.

3.1 Mutation strengths

The most successful self-adaptively controlled parameters are mutation parameters. This is a consequence of the direct influence of the mutation operator on the exploration behavior of the optimization algorithm: large mutation strengths cause large changes in solution space, decreasing

[8] Optimal progress is problem-dependent and can be stated theoretically on artificial functions [12].

mutation strengths allow an arbitrary approximation of the optimum, in particular in continuous solution spaces.

3.1.1 Continuous mutation strengths

In ES and EP self-adaptive strategy variables define the characteristics of mutation distributions. After recombination an individual $\mathbf{a}$ of a $(\mu \overset{+}{,} \lambda)$-ES with the N-dimensional objective variable vector $\mathbf{x} \in \mathbb{R}^N$ is mutated in the following way, i.e., for every solution $\mathbf{x}_1 \ldots \mathbf{x}_\lambda$ the ES applies the following equations:

$$\mathbf{x}' := \mathbf{x} + \mathbf{z}. \tag{13}$$

and

$$\mathbf{z} := (\sigma_1 \mathcal{N}_1(0,1), \ldots, \sigma_N \mathcal{N}_N(0,1)), \tag{14}$$

while $\mathcal{N}_i(0,1)$ delivers a Gaussian distributed number. The strategy parameter vector is mutated. A typical variation of σ that we will more closely get to know in Sect. 4.1.2 is the log-normal mutation:

$$\boldsymbol{\sigma}' := e^{(\tau_0 \mathcal{N}_0(0,1))} \cdot \left(\sigma_1 e^{(\tau_1 \mathcal{N}_1(0,1))}, \ldots, \sigma_N e^{(\tau_1 \mathcal{N}_N(0,1))}\right). \tag{15}$$

Note, that during the run of the algorithm, Eqs. 13, 14, and 15 are applied in reverse order: first, the mutation strengths are mutated, then the mutation is computed and added to each solution. As the mutation step size has an essential effect on the quality of the mutations, the stochastic search is often successful in controlling the step size successfully. Figure 3 shows an experimental comparison of an ES with constant mutation strengths, self-adaptive mutation strengths and the CMSA-ES of Sect. 1.3 on the *Sphere* problem with $N = 30$ dimensions. It illustrates the limited convergence behaviors of ES with constant mutation strengths. The experiments show: if the optimization algorithm is able to adapt to the changing fitness landscape online like the self-adaptive ES or the CMSA-ES, an unlimited approximation of the optimal solution is possible.

For a derivation of optimal step sizes σ^* in real-valued solution spaces, we refer to Beyer and Schwefel [15] and Beyer [12]. Further mutative parameters in real-valued solution spaces can be subject to self-adaptive control:

1. Asymmetric mutation: the skewness of the mutation distribution is controlled self-adaptively in the asymmetric mutation approach by Hildebrandt and Berlik [11, 35]. Although one of the classical assumptions for mutation operators is their unbiasedness [15], asymmetric mutation turns out to be advantageous on some problem classes, e.g., ridge functions.
2. Biased mutation: biased mutation by Kramer et al. [46] controls the center of the mutation ellipsoids self-adaptively shifting it by a bias vector $\mathbf{b}$. Biased mutation increases the success probabilities,[9] in particular in constrained solution spaces.
3. Correlated mutation: the correlated mutation operator by Schwefel [69] rotates the axes of the mutation ellipsoid—defining the space where Gaussian mutations are produced—to adapt to local success characteristics of the fitness landscape. Correlated mutations need $N(N-1)/2$ additional strategy parameters to rotate each mutation axis.
4. In Sect. 1.3 we got to know the CMSA-ES by Beyer [16] that evolves the covariance matrix $\mathbf{C}$ and the step sizes $\boldsymbol{\sigma}$ by means of self-adaptation. Similar to correlated mutation the matrix $\mathbf{C}$ defines rotation and scale of mutation axes and allows the adaptation to local fitness landscape characteristics.

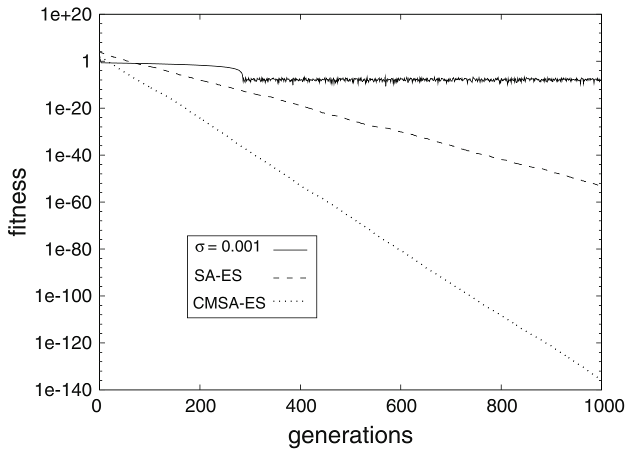

Fig. 3 Comparison of an ES with constant mutation strength ($\sigma = 0.001$), an ES with self-adaptive mutation strengths (SA-ES) and the CMSA-ES on the *Sphere* problem with $N = 30$. The self-adaptive SA-ES and CMSA-ES allow logarithmically linear approximation of the optimal solution

3.1.2 Discrete mutation strengths

Can self-adaptation also control discrete and combinatorial strategy parameters? Not many evolutionary algorithms with discrete strategy variables for combinatorial solution spaces exist. Punctuated crossover by Schaffer and Morishima [67] makes use of a bit string of discrete strategy variables that represent location and number of crossover points. Spears [74] introduced a similar approach for the selection of crossover operators. Kramer et al. [45] introduced self-adaptive crossover controlling the position of crossover points that are represented by integers, see Sect. 3.2.2.

Here, we review the approach to introduce a self-adaptive control of the *mutation strength* of inversion mutation

[9] The success probability p_s of an evolutionary operator is the relation of solutions with better fitness than the original solution $f(x)$ and all generated solutions.

for the traveling salesman problem (TSP) [43]. Inversion mutation is a famous example of a mutation operator for evolutionary algorithms on combinatorial TSP-like problems. It randomly chooses two edges of the TSP-solution, e.g. A_1A_2 and B_1B_2, and exchanges the connections resulting in A_1B_2 and B_1A_2. The number σ of successive applications of the inversion mutation operator can be treated as mutation strength. The inversion mutation operator can be equipped with self-adaptation: let $p_1, p_2 \in N^+$ be two randomly chosen points with $1 \le p_1 \le p_2 \le N$ and let $\mathbf{x}$ be a permutation of cities:

$$\mathbf{x} := (c_1, \ldots, c_{p_1}, c_{p_1+1}, \ldots, c_{p_2-1}, c_{p_2}, \ldots, c_N), \tag{16}$$

Inversion mutation inverses the part between p_1 and p_2 such that

$$\mathbf{x}' := \mathrm{INV}(\mathbf{x}) = (c_1, \ldots, c_{p_1}, c_{p_2-1}, \ldots, c_{p_1+1}, c_{p_2}, \ldots, c_N). \tag{17}$$

The number σ of successive applications of the inversion mutation operator can be treated as mutation strength. INV_σ denotes that the INV-operation is applied σ times. The idea of self-adaptive inversion mutation is to evolve σ self-adaptively. Let $\sigma_{t\text{-}1}$ be the number of successive INV-operations of the pervious generation. Then, σ_t is mutated $\sigma_t = \mathrm{mut}(\sigma_{t-1})$ and the inversion operator is repeated σ_t times on the objective variable vector $\mathbf{x}$. For mutation we recommend to apply meta-EP mutation with rounding, also see Sect. 4.1.1:

$$\sigma_t = \sigma_{t-1} + \mathrm{round}(\gamma \cdot \mathcal{N}(0,1)), \tag{18}$$

or Bäck's and Schütz's continuous log-normal operator, see Sect. 4.1.3. An experimental analysis shows that self-adaptive inversion mutation is able to speed up the optimization process, in particular at the beginning of the search: Fig. 4 shows the fitness development of 100 generations and the development of strategy parameter σ averaged over 25 runs. Self-adaptive inversion mutation and static inversion mutation are compared on problem *gr666* from the library of TSP-problems by Reinelt [63]. Parameter σ is increasing at the beginning and decreasing during the optimization as fewer and fewer edge swaps are advantageous to improve and not to destroy good solutions. But, when reaching the border of the strategy parameter $\sigma = 1$, the search slows down, as self-adaptation still produces solutions with $\sigma > 1$ resulting in a worse success rate. This problem can be solved by setting $\sigma = 1$ constantly if the majority of the offspring population, i.e., more than $\lambda/2$, is voting for this setting.

3.2 Crossover properties

Crossover is the second main operator in evolutionary computation. But its role has not been satisfactorily explained. Is it an exploration operator, i.e., creating noise,

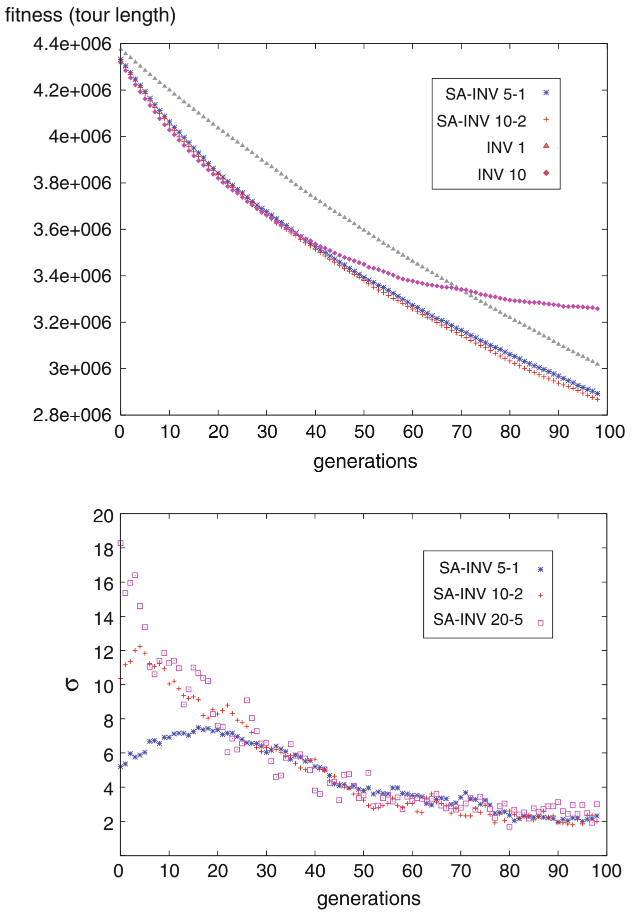

Fig. 4 Self-adaptive control of discrete mutation strengths on the combinatorial problem *gr666* from the library of TSP-problems by Reinelt [63]. Upper part: average fitness development of self-adaptive inversion mutation (SA-INV) and INV over 25 runs. $\text{SA-INV}_{5\text{-}1}$ and $\text{SA-INV}_{10\text{-}2}$ (the first number indicates the initial σ, the second number indicates γ) are the fastest algorithms, at least until t = 100.INV_{10} (with 10 successive inversion mutations for each solution) suffers from premature fitness stagnation. Lower part: Average development of σ—also for $\text{SA-INV}_{10\text{-}2}$—over 25 runs. The mutation strength σ is increasing at the beginning and adapting to the search process conditions during the evolutionary process

or an exploitation operator identifying and arranging building blocks? Are there optimal crossover points and, if they exist, can they be found self-adaptively? The building block hypothesis (BBH) by Goldberg [27] and Holland [36] assumes that crossover combines *different* useful blocks of parents. The genetic repair effect (GR) [12, 15] hypothesis assumes that *common* properties of parental solutions are mixed. Most of today's crossover operators do not exhibit adaptive properties, but are either completely random or fixed. Nevertheless, self-adaptive approaches exist.

3.2.1 *Crossover probabilities*

A step before the identification of crossover points concerns the question, how often the crossover operator should be applied. In many evolutionary algorithms, crossover is applied with a fixed crossover probability p_c. In case of ES, recombination is applied for every solution, i.e., $p_c = 1.0$. Davis [18] states that it can be convincingly argued that these probabilities should vary over the course of the optimization run. Also Maruo et al. [49] introduced an approach that controls probability p_c self-adaptively. The authors had the intention to build a flexible parameter-free evolutionary algorithm. They reported successful results on combinatorial benchmark problems. The tests in Sect. 3.1.2 have shown that success probabilities in combinatorial solution spaces decrease significantly during the approximation of the optimum. The constructive effect of crossover is the reason for the increase of its probability at the beginning. Its disruptive effect in later sensitive situations may be the reason that the self-adaptive process "fades out" crossover. A self-adaptive disruptive operator will decrease its application probability during the course of evolution.

3.2.2 *Crossover points*

In eukaryotes DNA regions exist that are more frequently subject to be broken off during the meiose than others. These regions are called *meiotic recombination hotspots*. Can parameters like crossover probabilities be controlled by means of self-adaptation? The first attempt to integrate self-adaptation into structural properties of crossover was *punctuated crossover* by Schaffer and Morishima [67]. Punctuated crossover adapts the crossover points for bit string representations and was experimentally proven better than standard one-point crossover. A strategy parameter vector with the same length as the objective variable vector is added to the individual's representation. It is filled with ones—the so called punctuation marks—indicating, where recombination should split up the solutions. Punctuation marks from both parents mark, from which parent the next part of the genome has to be copied to the offspring. This self-adaptive variant outperforms a canonical genetic algorithm on four of De Jong's five test functions [67]. Spears [74] argues that self-adaptation is not responsible for the success of punctuated crossover, but the flexibility of using more than one crossover point. Another technique refers to the identification of building blocks, the LEGO (linkage evolving genetic operator) algorithm [72]. Each gene exhibits two bits indicating the linkage to neighbored genes, which is only possible if the same bits are set. New candidate solutions are created from left to right conducting tournaments between parental blocks. Linkage is an idea that stems from biology. It describes genes located close to each other on the same chromosome that have a greater chance than average of being inherited from the same parent. Harik et al. [30] define linkage as the probability of two variables being inherited from the same parent. Later, Harik et al. [31] analyzed the relationship between the linkage learning problem and that of learning probability distributions over multi-variate spaces. They argue that both problems are equivalent and propose algorithmic schemes from this observation. Meyer-Nieberg and Beyer [52] point out that even the standard 1- or n-point crossover operators for bit string representations exhibit properties of a self-adaptive mutation operator. Bit positions that are common in both parents are transferred to the offspring individuals, while other positions are filled randomly. This assumption is consistent with the GR hypothesis, and in contrast to the BBH.

Table 1 Experimental comparison of 1-point and SA-1-point crossover on OneMax and 3-SAT

	Best	Worst	Mean	Dev
OneMax				
1-point	16	93	28.88	22.3
SA-1-point	13	71	27.68	19.7
3-SAT				
1-point	15	491	63.68	291
SA-1-point	7	498	62.72	264

The figures show the number of generations until the optimum is found. No significant superiority of any of the two 1-point variants can be reported in mean, but the self-adaptive variants achieve the best results

Now, we test self-adaptive 1-point crossover experimentally. One crossover point σ is chosen randomly with uniform distribution. The left part of the bit-string from the first parent is combined with the right part of the second parent, and in turn for a second solution. For the self-adaptive extension of 1-point crossover (SA-1-point) the strategy part of the chromosome is extended by an integer value $\Sigma = (\sigma)$ with $\sigma \in [1, N-1]$ determining the crossover point. The crossover point σ is the position between the σ-th and the $(\sigma+1)$-th element. Given two parents $\mathbf{p}^1 = (p_1^1, \ldots, p_N^1, \sigma^1)$ and $\mathbf{p}^2 = (p_1^2, \ldots, p_N^2, \sigma^2)$ self-adaptive 1-point crossover selects the crossover point σ^* of the parent with higher fitness and creates the two offspring solutions:

$$\mathbf{o}^1 := (p_1^1, \ldots, p_{\sigma^*}^1, p_{(\sigma^*+1)}^2, \ldots, p_N^2, \sigma^*), \tag{19}$$

and

$$\mathbf{o}^2 := (p_1^2, \ldots, p_{\sigma^*}^2, p_{(\sigma^*+1)}^1, \ldots, p_N^1, \sigma^*). \tag{20}$$

Table 1 shows the results of standard 1-point crossover and SA-1-point crossover on OneMax[10] and on 3-SAT[11].

[10] OneMax is a pseudo-binary function. For a bit string $\mathbf{x} \in \{0,1\}^N$, maximize $f(\mathbf{x}) := \sum_{i=1}^N x_i$ with $\mathbf{x} \in \{0,1\}^N$ optimum $\mathbf{x}^* = (1, \ldots, 1)$ with $f(\mathbf{x}^*) = N$.

[11] 3-SAT is a propositional satisfiability problem, each clause contains $k = 3$ literals; we randomly generate formulas with $N = 50$ variables and 150 clauses; the fitness function is: $\frac{\#\text{of true clauses}}{\#\text{of all clauses}}$.

We use $\mu = 150$ as population size and the mutation probabilities $p_m = 0.001$ for OneMax and $p_m = 0.005$ for 3-SAT. For further experimental details we refer to [45]. The strategy part σ^* may be encoded as an integer or in binary representation as well. In order to allow self-adaptation, the crossover point σ^* has to be mutated. For this purpose meta-EP mutation, see Sect. 4.1.1, with a rounding procedure is used. The numbers show the generations until the optimum is found. Although the mean of SA-1-point lies under the mean of 1-point crossover, the standard deviation is too high to assert statistical significance. SA-1-point was able to find the optimum of 3-SAT in only 7 generations, which is very fast. But it also needs the longest time with the worst result. We observe no statistically significant improvements of the self-adaptive variants in comparison to standard operators.

3.3 Self-Adaptive operator selection

The self-adaptive control of crossover probabilities has shown that operators can be faded in and faded out. A self-adaptive switch between various operators seems to be a further logical step. Spears [74] proposed one-bit self-adaptive crossover that makes use of a bit indicating the crossover type. The bit switches between uniform and two-point crossover. From his tests Spears derived the conclusion that the key feature of his self-adaptation experiment is not the possibility to select the best operator, but the degree of freedom for the evolutionary algorithm to choose among two operators during the search. Our own tests of a similar selection heuristic for mutation operators in continuous solution spaces revealed that self-adaptive operators adapt fast to the optimization process and outperform operators that did not have the chance to adapt their parameterizations, also if the latter are potentially better operators.

3.4 Global parameters

Global strategy variable vectors parameterize global properties of the evolutionary technique like population sizes or selection operator parameters. As strategy parameters are part of each individual, the local individual-level information must be aggregated. Eiben, Schut and Wilde [21] propose a method to control the population size and the selection pressure using tournament selection self-adaptively. The idea of the approach is to aggregate individual-level strategy values to global values. As usual, the local strategy parameters are still part of the individuals' genomes and participate in recombination and mutation. They introduce an aggregation mechanism that sums up all votes of the individuals for the global parameter. Summing up strategy values is equal to computing the average.

Table 2 Experimental comparison of the CMSA-ES with two variants that control the global parameter τ_c and μ self-adaptively

	Best	Worst	Mean	Dev
CMSA-ES	8,687	10,303	9,486.9	426.4
CMSA-ES (τ_c)	11,212	27,170	17,708.3	3,786.1
CMSA-ES (μ)	8,180	13,394	9,730.1	1,457.4

The best results have been accomplished with the self-adaptive parental population size

We investigate the approach of self-adaptive global parameters for the covariance matrix parameter τ_c and the population sizes μ and λ of the CMSA-ES introduced in Sect. 1.3. First, every individual $\mathbf{a}_i$ gets a strategy parameter σ_i determining a *vote* for τ_c. At the end of each generation t the information on individual-level is aggregated to a global parameter:

$$\tau_c := \frac{1}{\mu}\sum_{i=1}^{\mu}\sigma_i. \qquad (21)$$

We tested the approach on the *Sphere* problem with $N = 30$, $\mu = 25$ and $\lambda = 100$. Table 2 shows the experimental results, i.e., the number of fitness function calls until the optimum is reached with an accuracy of 10^{-10}. Obviously, controlling τ_c by means of self-adaptation is possible—as the optimum is found—although a slight deterioration is observed. By varying the initial setting for τ_c, it can be observed that a self-adaptive control allows to escape from ill-conditioned initial settings and leads to robustness against bad initializations. In a second experiment we used the same mechanism to evolve the population size μ equipping every individual $\mathbf{a}_i$ with the corresponding strategy parameter. For this experiment we set $\lambda = 4 \cdot \mu$, starting from $\mu = 25$ and using $\tau_c = 30$. We restrict the population size to the interval $\mu \in [2, 250]$. It turns out that self-adaptation decreases the population size fast to the minimum of $\mu = 2$. The process accelerates, in particular at the beginning of the search. Nevertheless, not every run is accelerated, but the process may also lead to a deterioration, resulting in a high standard deviation. In multimodal fitness landscapes we observed that a small population often gets stuck in local optima. But then, the algorithm increases the population size—as a high number of offspring increases the success probability p_s. We can conclude that self-adaptation of global parameters is reasonable to correct bad initializations. Whether it leads to an improvement or not depends on the relevance of the strategy parameter.

4 On search in the strategy parameter space

Important for successful self-adaptation is the choice of appropriate exploration operators for the search in the strategy parameter space. For this purpose various mutation operators have been introduced. Not much effort has been spent on crossover operators for strategy variables so far. But any crossover operators that are known for objective variables may be applied. We draw special attention to multi-recombination as it allows an efficient aggregation of successful endogenous strategy parameters and is strongly related to the aggregation of global strategy parameters.

4.1 Mutation operators for strategy parameters

4.1.1 Meta-EP mutation operator

The *Meta-EP* mutation operator—simply adding Gaussian noise to σ—is a classical variant that has first been used to adapt the angles of the correlated mutation in ES [69]:

$$\sigma' := \sigma + \tau \cdot \mathcal{N}(0,1). \tag{22}$$

The learning parameter τ defines the "mutation strength of the mutation strengths". It is a problem-dependent parameter, for which various settings have been proposed, e.g., $\tau \approx 5°$ for the angles of correlated mutations. Adding Gaussian noise is a quite natural way of producing mutations. We have already applied meta-EP mutation for the adaptation of the inversion mutation in Sect. 3.1.2.

If the strategy parameter is discrete, rounding procedures have to discretize the real-valued noise. Furthermore, many strategy parameters may be restricted to intervals, e.g., $\sigma \geq 0$. For this purpose, in particular to scale logarithmically in $\mathbb{R}^+$, log-normal mutation is a better choice.

4.1.2 Schwefel's Continuos Log-Normal mutation operator

The log-normal mutation operator by Schwefel [68] has become famous for the mutation of step sizes in real-valued search domains, i.e., in $\mathbb{R}^N$. It is based on multiplication with log-normally distributed random numbers:

$$\sigma' := \sigma \cdot e^{\tau \cdot \mathcal{N}(0,1)}. \tag{23}$$

Strategy parameter σ cannot be negative and scales logarithmically between values close to 0 and infinity, i.e., high values with regard to the used data structure. Again, the problem dependent learning rate τ has to be set adequately. For the mutation strengths of ES on the continuous *Sphere* model, theoretical investigations [15] lead to the optimal setting:

$$\tau := \frac{1}{\sqrt{N}}. \tag{24}$$

This setting may not be optimal for all problems and further parameter tuning is recommended. A more flexible approach is to mutate each of the N dimensions independently:

$$\boldsymbol{\sigma}' := e^{(\tau_0 \mathcal{N}_0(0,1))} \cdot \left(\sigma_1 e^{(\tau_1 \mathcal{N}_1(0,1))}, \ldots, \sigma_N e^{(\tau_1 \mathcal{N}_N(0,1))}\right), \tag{25}$$

and

$$\tau_0 := \frac{c}{\sqrt{2N}}, \tag{26}$$

as well as

$$\tau_1 := \frac{c}{\sqrt{2\sqrt{N}}}. \tag{27}$$

Setting parameter $c = 1$ is a recommendable choice. Kursawe [47] analyzed parameters τ_0 and τ_1 using a nested ES on various test problems. His analysis shows that the choice of mutation parameters is problem-depended.

4.1.3 Continuous interval Log-Normal mutation operator

As step sizes for Gaussian mutations might scale from values near zero to a maximum value with regard to the used data structure, other strategy variables like mutation probabilities are restricted to the interval between 0 and 1, i.e., $\sigma \in (0, 1)$. To overcome this problem Bäck and Schütz introduced the following operator [10]:

$$\sigma' := \left(1 + \frac{1-\sigma}{\sigma} \cdot e^{\tau \cdot \mathcal{N}(0,1)}\right)^{-1} \tag{28}$$

The parameter τ controls the magnitude of mutations similar to τ of Schwefel's log-normal operator. Eiben makes use of this operator to adapt global parameters like selection pressure and population sizes [21].

4.2 Crossover for strategy parameters

For the recombination of strategy parameters any of the crossover operators for objective variables is applicable, e.g., 1- or N-point crossover for discrete representations. Arithmetic recombination, i.e., the arithmetic mean, is a good choice for real-valued strategy parameters [15, 60, 68]. If ρ is the number of parents that are in most cases selected with uniform distribution, σ is the average of the ρ strategy variables:

$$\sigma := \frac{1}{\rho}\sum_{i=1}^{\rho} \sigma_i. \tag{29}$$

But also geometric recombination may be applied:

$$\sigma := \left(\prod_{i=1}^{\rho} \sigma_i\right)^{1/\rho}. \tag{30}$$

A special case concerns the case of multi-recombination, i.e., $\rho = \mu$. It allows a fast and easy implementation with two advantages. First, the strategy parameters do not have to be inherited to the next generation, but can be *aggregated* at the end of each iteration. In the next generation from this single *aggregated* strategy parameter λ variations are produced, bound to each individual $\mathbf{a}_i$, then μ are selected and aggregated, etc. Second, this implementation allows to save copy operations: if the length N of an individual is very large, not the whole objective or strategy parameter has to be copied, but only one main individual $\mathbf{x}$ and a list of changes. Of course, this trick can also be applied to both objective and strategy variables.

5 Premature convergence

The phenomenon that the evolutionary search converges, but not towards the optimal solution, is called premature convergence. Premature convergence of the search process due to a premature decrease of mutation strengths belongs to the most frequent problems of self-adaptive mutation strength control. Evolution rewards short term success: the evolutionary process can get stuck in local optima. Premature convergence is a result of a premature mutation strength reduction or a decrease of variance in the population. The problem is well known and experimentally proven. Only few theoretical works concentrate on this phenomenon, e.g., the work of Rudolph [65] who investigates premature convergence of self-adaptive mutation strength control. He assumes that the (1+1)-EA is located in the vicinity $\mathcal{P}$ of a local solution with fitness $f = \varepsilon$. He proves that the probability to move from a solution in this set $\mathcal{P}$ to a solution in a set of solutions $\mathcal{S}$ with a fitness f less than ε (which in particular contains the global optimum, the rest of the search domain exhibits fitness values $f > \varepsilon$) is smaller than 1. This result even holds for an infinite time horizon. Hence, an evolutionary algorithm can get stuck at a non-global optimum with a positive probability. Rudolph also proposes ideas to overcome this problem.

Small success probabilities p_s may result from disadvantageous fitness landscape and operator conditions. An example for such disadvantageous conditions are constrained real-valued solution spaces [42]. Many numerical problems are constrained. Whenever the optimum lies in the vicinity of the constraint boundary, the success rates decrease and make successful random mutations almost impossible. Low success rates result in premature step size reduction and finally lead to premature convergence. Table 3 shows the corresponding result of a (25,100)-CMSA-ES on Schwefel's problem[12] 2.40 and the tangent

Table 3 Experimental results of the CMSA-ES using death penalty (DP) on the two constrained problems TR2 and 2.40

CMSA-ES /DP	Best	Mean	Worst	Dev	ffe
TR2	3.1×10^{-7}	2.1×10^{-4}	1.1×10^{-3}	3.8×10^{-4}	10,120
2.40	31.9	127.4	280.0	65.2	40,214

problem[13] TR2. The termination condition is fitness stagnation: the algorithm terminates, if the fitness gain from generation t to generation $t+1$ falls below $\theta = 10^{-12}$ in successive generations. In this case the magnitude of the step sizes is too small to effect further improvements. Parameters *best*, *mean*, *worst* and *dev* describe the achieved fitness[14] of 25 experimental runs while *ffe* counts the average number of fitness function evaluations. The results show that the CMSA-ES with death penalty (discarding infeasible solutions in each generation until enough feasible ones are generated) is not able to approximate the optimum of the problem satisfactorily, but shows relatively high standard deviations *dev*. To avoid this problem the application of constraint handling methods is recommended, e.g., meta-modeling of the constraint boundary, see Kramer et al. [44].

The phenomenon of premature step size reduction at the constraint boundary has been analyzed—in particular for the condition that the optimum lies on the constraint boundary or even in a vertex of the feasible search space [42]. In such cases the evolutionary algorithm frequently suffers from low success probabilities near the constraint boundaries. Under simple conditions, i.e., a linear objective function, linear constraints and a comparably simple mutation operator, the occurrence of premature convergence due to a premature decrease of step sizes was proven. This phenomenon occurs in continuous search spaces if the coordinate system that is basis of mutation is not aligned to the constraint boundary. The region of success is cut off by the constraint and decreases the success probability in comparison to the case of small step sizes that are not cut off. Also Arnold et al. [2] analyzed the behavior at the boundary of linear constraints and models the distance between the search point and the constraint boundary with a Markov chain. Furthermore, they discuss the working of step length adaptation mechanisms based on success probabilities.

The following causes for premature convergence could be derived. Stone and Smith [75] came to the conclusion

[12] Schwefel's problem 2.40 [69]: Minimize $f(\mathbf{x}) := -\sum_{i=1}^{5} x_i$, constraints $g_j(\mathbf{x}) := x_j \geq 0$, for $j = 1, \ldots, 5$ and $g_j(\mathbf{x}) = -\sum_{i=1}^{5} (9+i)x_i + 50000 \geq 0$, for $j = 6$, minimum $\mathbf{x}^* = (5000, 0, 0, 0, 0)^T$ with $f(\mathbf{x}^*) = -5000$.

[13] TR (tangent problem): minimize $f(\mathbf{x}) := \sum_{i=1}^{N} x_i^2$, constraints $g(\mathbf{x}) := \sum_{i=1}^{N} x_i - t > 0$, $t \in \mathbb{R}$ (tangent), for N=k and t=k the minimum lies at: $\mathbf{x}^* = (1, \ldots, 1)^T$, with $f(\mathbf{x}^*) = k$, TR2 means $N = 2$ and $t = 2$.

[14] difference between the optimum and the best solution $|f(\mathbf{x}^*) - f(\mathbf{x}^{best})|$.

that low innovation rates[15] and high selection pressure result in low diversity. They investigated Smith's discrete self-adaptation genetic algorithm on multimodal functions. Another reason for premature convergence was revealed by Liang et al. [48]. They point out that a solution with a high fitness, but a far too small step size in one dimension is able to cause stagnation by inheriting this mutation strength to all descendants. As the mutation strength changes with a successful mutation according to the principle of self-adaptation, Liang et al. [48] considered the probability that after k successful mutations the step size is smaller than an arbitrarily small positive number ε. This results in a loss of step size control of their (1+1)-EP. Meyer-Nieberg and Beyer [52] point out that the reason for the premature convergence of the EP optimization algorithm could be that the operators do not fulfill the postulated requirements of mutation operators. Hansen [29] examined the conditions when self-adaptation fails, in particular the inability of the step sizes to increase. He tries to answer the question, whether a step size increase is affected by a bias of the genetic operators or due to the link between objective and strategy parameters. Furthermore, he identifies two properties of an evolutionary algorithm: first, the descendants' object parameter vector should be point-symmetrically distributed after recombination and mutation. Second, the distribution of the strategy parameters given the object vectors after recombination and mutation should be identical for all symmetry pairs around the point-symmetric center.

6 Conclusion and outlook

The online-control of strategy variables in evolutionary computation plays an essential role for successful search processes. Various parameter control techniques have been proposed within the last decades. Recent developments like the CMSA-ES show that self-adaptation is still developing over four decades after the idea has been introduced for the first time. Self-adaptation is an efficient way to control the strategy parameters of an evolutionary optimization algorithm automatically during optimization. It is based on implicit evolutionary search in the space of strategy parameters, and has been proven well as online parameter control method for a variety of strategy parameters, from local to global ones. For each type of strategy parameter adequate operators for the exploration of the strategy variable search space have been proposed. The best results for self-adaptive parameters have been achieved for mutation operators. Most theoretical work on self-adaptation concentrates on mutation. A necessary condition for the success of self-adaptation is a tight link between strategy parameters and fitness: if the quality of the search process strongly depends on a particular strategy variable, self-adaptive parameter control is reasonable. But not every parameter type offers a strong link. e.g., crossover points suffer from a *weak link*.

Although self-adaptation is a rather robust online parameter control technique, it may fail if the success probability in the solution space is low, e.g., at the boundary of constraints. Premature convergence can be avoided by specialized methods that increase the success probabilities like constraint handling methods [44]. A further possibility is to define lower bounds on the strategy variables. But the drawback is that a lower bound on the strategy parameters may prevent convergence to important values. An increase of the number of offspring solutions is another possibility to decrease the probability of unsuccessful strategy settings, but has to be paid with more function evaluations. In some cases tuning of the strategy parameters, e.g., τ in the case of Gaussian mutations for ES, see Sect. 4.1.2, might lead to an increase of success probabilities and is a solution to the problem of premature convergence. Further research on the conditions that allow successful self-adaptation will be necessary. In the future self-adaptation might be the key property towards a parameter-free evolutionary optimization algorithm.

Acknowledgments The author thanks Günter Rudolph and the anonymous reviewers for their helpful comments to improve the manuscript.

References

1. Angeline PJ (1995) Adaptive and self-adaptive evolutionary computations. In: Palaniswami M, Attikiouzel Y (eds) Computational intelligence a dynamic systems perspective. IEEE Press, New York, pp 152–163
2. Arnold DV, Brauer D (2008) On the behaviour of the (1+1)-ES for a simple constrained problem. In: Proceedings of the 10th conference on parallel problem solving from nature—PPSN X, pp 1–10
3. Auger A (2003) Convergence results for (1, λ)-SA-ES using the theory of ϕ-irreducible markov chains. In: Proceedings of the evolutionary algorithms workshop of the 30th international colloquium on automata, languages and programming
4. Bäck T (1998) An overview of parameter control methods by self-adaption in evolutionary algorithms. Fundam Inf 35(1–4):51–66
5. Bartz-Beielstein T (2006) Experimental research in evolutionary computation: the new experimentalism. Natural computing series. Springer, April
6. Bartz-Beielstein T, Lasarczyk C, Preu M (2005) Sequential parameter optimization. In: McKay B, et al (eds) Proceedings of the IEEE congress on evolutionary computation—CEC, vol 1. IEEE Press, pp 773–780

[15] Low innovation rates are caused by variation operators that produce offspring not far away from their parents, e.g. by low mutation rates.

7. Bartz-Beielstein T, Preuss M (2006) Considerations of budget allocation for sequential parameter optimization (SPO). In: Paquete L, et al. (eds) Workshop on empirical methods for the analysis of algorithms, proceedings, Reykjavik, Iceland, pp 35–40
8. Bäck T (1991) Self-adaptation in genetic algorithms. In: Proceedings of the 1st European conference on artificial life—ECAL, pp 263–271
9. Bäck T (1992) The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Proceedings of the 2nd conference on parallel problem solving from nature—PPSN II, pp 85–94
10. Bäck T, Schütz M (1996) Intelligent mutation rate control in canonical genetic algorithms. In: Foundation of intelligent systems, 9th international symposium, ISMIS '96. Springer, pp 158–167
11. Berlik S (2004) A step size preserving directed mutation operator. In: Proceedings of the 6th conference on genetic and evolutionary computation—GECCO, pp 786–787
12. Beyer H-G (2001) The theory of evolution strategies. Springer, Berlin
13. Beyer H-G, Deb K (2001) On self-adaptive features in real-parameter evolutionary algorithms. IEEE Trans Evol Comput 5(3):250–270
14. Beyer H-G, Meyer-Nieberg S (2006) Self-adaptation on the ridge function class: first results for the sharp ridge. In: Proceedings of the 9th conference on parallel problem solving from nature—PPSN IX, pp 72–81
15. Beyer HG, Schwefel HP (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1:3–52
16. Beyer HG, Sendhoff B (2008) Covariance matrix adaptation revisited—the cmsa evolution strategy. In: Proceedings of the 10th conference on parallel problem solving from nature—PPSN X, pp 123–132
17. Coello CA (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127
18. Davis L (1989) Adapting operator probabilities in genetic algorithms. In: Proceedings of the 3rd international conference on genetic algorithms, San Francisco, Morgan Kaufmann Publishers Inc, pp 61–69
19. de Landgraaf W, Eiben A, Nannen V (2007) Parameter calibration using meta-algorithms. In: Proceedings of the IEEE congress on evolutionary computation—CEC, pp 71–78
20. DeJong K (2007) Parameter setting in EAs: a 30 year perspective. In: Parameter setting in evolutionary algorithms, studies in computational intelligence. Springer, pp 1–18
21. Eiben A, Schut MC, de Wilde A (2006) Is self-adaptation of selection pressure and population size possible? A case study. In: Proceedings of the 9th conference on parallel problem solving from nature—PPSN IX, pp 900–909
22. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141
23. Eiben AE, Michalewicz Z, Schoenauer M, Smith JE (2007) Parameter control in evolutionary algorithms. In: Parameter setting in evolutionary algorithms, studies in computational intelligence. Springer, pp 19–46
24. Fogarty TC (1989) Varying the probability of mutation in the genetic algorithm. In: Proceedings of the 3rd international conference on genetic algorithms, San Francisco, Morgan Kaufmann Publishers Inc, pp 104–109
25. Fogel DB, Fogel LJ, Atma JW (1991) Meta-evolutionary programming. In: Proceedings of 25th asilomar conference on signals, systems & computers, pp 540–545
26. georg Beyer H, Arnold DV (2003) Qualms regarding the optimality of cumulative path length control in csa/cma-evolution strategies. Evol Comput 11
27. Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, Reading
28. Grefenstette J (1986) Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 16(1):122–128
29. Hansen N (2006) An analysis of mutative sigma self-adaptation on linear fitness functions. Evol Comput 14(3):255–275
30. Harik GR, Goldberg DE (1997) Learning linkage. In: Foundations of genetic algorithms 4. Morgan Kaufmann, pp 247–262
31. Harik GR, Lobo FG, Sastry K (2006) Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In: Scalable optimization via probabilistic modeling, studies in computational intelligence, Springer, pp 39–61
32. Herdy M (1992) Reproductive isolation as strategy parameter in hierarchically organized evolution strategies. In: Proceedings of the 10th conference on parallel problem solving from nature—PPSN II, pp 207–217
33. Hesser J, Männer R (1990) Towards an optimal mutation probability for genetic algorithms. In: Proceedings of the 10th conference on parallel problem solving from nature—PPSN I, London, UK, Springer-Verlag, pp 23–32
34. Hesser J, Männer R (1992) Investigation of the m-heuristic for optimal mutation probabilities. In PPSN, pp 115–124
35. Hildebrand L (2002) Asymmetrische evolutionsstrategien. PhD thesis, University of Dortmund
36. Holland JH (1992) Adaptation in natural and artificial systems, 1st edn, MIT Press, Cambridge
37. Jägersknpper J (2005) Rigorous runtime analysis of the (1+1) es: 1/5-rule and ellipsoidal fitness landscapes. In: Proceedings of the workshop on foundation of genetic algorithms FOGA, pp 260–281
38. Jägersknpper J (2006) Probabilistic runtime analysis of $(1 + \lambda)$es using isotropic mutations. In: Proceedings of the 8th conference on genetic and evolutionary computation—GECCO, New York, ACM, pp 461–468
39. Joines J, Houck C (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Fogel DB (eds) Proceedings of the 1st IEEE conference on evolutionary computation, Orlando, Florida, IEEE Press, pp 579–584
40. Jong KAD (1975) An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan
41. Koumoutsakos P, Muller SD (2006) Flow optimization using stochastic algorithms. Lecture Notes Control Inf Sci 330:213–229
42. Kramer O (2008) Premature convergence in constrained continuous search spaces. In: Proceedings of the 10th conference on parallel problem solving from nature—PPSN X, Berlin, Springer, to appear
43. Kramer O (2008) Self-adaptive inversion mutation for combinatorial representations. In: Proceedings of the 2008 international conference on genetic and evolutionary methods, to appear
44. Kramer O, Barthelmes A, Rudolph G (2009) Surrogate constraint functions for cma evolution strategies. In: Proceedings of the conference on artificial intelligence and automation, page to appear
45. Kramer O, Koch P (2007) Self-adaptive partially mapped crossover. In: Proceedings of the 9th conference on genetic and evolutionary computation—GECCO, New York, ACM Press, pp 1523–1523
46. Kramer O, Ting CK, Büning HK (2005) A new mutation operator for evolution strategies for constrained problems. In: Proceedings of the IEEE congress on evolutionary computation—CEC, pp 2600–2606
47. Kursawe F (1999) Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien—Metastrategien. PhD thesis, University of Dortmund

48. Liang KH, Yao X, Liu Y, Newton CS, Hoffman D (1998) An experimental investigation of self-adaptation in evolutionary programming. In: Proceedings of the 7th international conference on evolutionary programming VII—EP, Berlin, Springer, pp 291–300
49. Maruo MH, Lopes HS, Delgado MR (2005) Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. In: Proceedings of EvoCOP, pp 154–165
50. Mercer RE, Sampson JR (1978) Adaptive search using a reproductive metaplan. Kybernetes 7:215–228
51. Mersch B, Glasmachers T, Meinicke P, Igel C (2006) Evolutionary optimization of sequence kernels for detection of bacterial gene starts. In: ICANN (2), pp 827–836
52. Meyer-Nieberg S, Beyer HG (2007) Self-adaptation in evolutionary algorithms. In: Lobo FG, Lima CF, Michalewicz Z (eds) Parameter setting in evolutionary algorithms. Springer, Berlin
53. Mezura-Montes E, Palomeque-Ortiz AG (2009) Self-adaptive and deterministic parameter control in differential evolution for constrained optimization. Constraint-Handl Evol Optim 189:95–120
54. Mühlenbein H (1992) How genetic algorithms really work: mutation and hillclimbing. In: Proceedings of the 2nd conference on parallel problem solving from nature—PPSN II, pp 15–26
55. Nannen V, Eiben A (2006) A method for parameter calibration and relevance estimation in evolutionary algorithms. In: Proceedings of the 8th conference on genetic and evolutionary computation—GECCO, New York, ACM Press, pp 183–190
56. Nannen V, Eiben A (2007) Relevance estimation and value calibration of evolutionary algorithm parameters. In: IJCAI, pp 975–980
57. Ostermeier A, Gawelczyk A, Hansen N (1994) A derandomized approach to self adaptation of evolution strategies. Evol Comput 2(4):369–380
58. Ostermeier A, Gawelczyk A, Hansen N (1995) A derandomized approach to self adaptation of evolution strategies. Evol Comput 2(4):369–380
59. Preuss M, Bartz-Beielstein T (2007) Sequential parameter optimization applied to self-adaptation for binary-coded evolutionary algorithms. In: Parameter setting in evolutionary algorithms, studies in computational intelligence. Springer, pp 91–119
60. Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart
61. Rechenberg I (1994) Evolutionsstrategie ’94. Frommann-Holzboog, Stuttgart
62. Reed J, Toombs R, Barricelli NA (1967) Simulation of biological evolution and machine learning: I. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. J Theor Biol 17:319–342
63. Reinelt G (1991) Tsplib—a traveling salesman problem library. ORSA J Comput 3:376–384
64. Rosenberg RS (1967) Simulation of genetic populations with biochemical properties. PhD thesis, University of Michigan
65. Rudolph G (2001) Self-adaptive mutations may lead to premature convergence. IEEE Trans Evol Comput 5(4):410–414
66. Schaffer JD, Caruana R, Eshelman LJ, Das R (1989) A study of control parameters affecting online performance of genetic algorithms for function optimization. In: Proceedings of the 3rd international conference on genetic algorithms—ICGA 1989, pp 51–60
67. Schaffer JD, Morishima A (1987) An adaptive crossover distribution mechanism for genetic algorithms. In: Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application, Hillsdale, NJ, USA, L. Erlbaum Associates Inc, pp 36–40
68. Schwefel H-P (1974) Adaptive Mechanismen in der biologischen Evolution und ihr Einflus auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut fnr Mess- und Regelungstechnik, TU Berlin
69. Schwefel HP (1995) Evolution and Optimum Seeking. Sixth-generation computer technology. Wiley Interscience, New York
70. Semenov MA, Terkel DA (2003) Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic lyapunov function. Evol Comput 11(4):363–379
71. Smith J (2001) Modelling GAs with self adaptive mutation rates. In: Proceedings of the genetic and evolutionary computation conference, pp 599–606
72. Smith J, Fogarty TC (1996) Recombination strategy adaptation via evolution of gene linkage. In: Proceedings of the IEEE congress on evolutionary computation—CEC, pp 826–831
73. Smith J, Fogarty TC (1996) Self adaptation of mutation rates in a steady state genetic algorithm. In: Proceedings of the international conference on evolutionary computation—ICEC, pp 318–323
74. Spears WM (1995) Adapting crossover in evolutionary algorithms. In: McDonnell JR, Reynolds RG, Fogel DB (eds) Proceedings of the fourth annual conference on evolutionary programming, Cambridge, MIT Press, pp 367–384
75. Stone C, Smith J (2002) Strategy parameter variety in self-adaptation of mutation rates. In: Proceedings of the genetic and evolutionary computation conference—GECCO, San Francisco, Morgan Kaufmann Publishers Inc, pp 586–593
76. Weinberg R (1970) Computer simulation of a living cell. PhD thesis, University of Michigan