# Variable Metrics in Evolutionary Computation

Nikolaus Hansen – INRIA

September 2009

# Contents

**Abstract**

This thesis considers variable metrics in the context of stochastic, function-value free optimization in continuous search spaces. We argue that the choice of a (variable) metric or equivalently the choice of a coordinate system can be decoupled from the underlying optimization procedure. An *adaptive encoding* procedure is presented, that is in principle applicable to any optimization procedure, and is proved to recover the *covariance matrix adaptation evolution strategy* (CMA-ES) when applied to a simple isotropic *evolution strategy* with step-size adaptation. The proof suggests that adaptive encoding should be able to improve the performance of many stochastic optimization algorithms in particular on ill-conditioned, non-separable objective functions.

# 1 Introduction

*Evolutionary Computation* (EC) or *Evolutionary Algorithms* (EAs) is a collection of meta-heuristics that borrow concepts from the biological evolution and have become popular in the last decades. In this work we consider EC in the context of **stochastic search** or optimization. In a search or optimization problem we consider an objective, cost, loss, error, or fitness function

$$f : S \to \mathbb{R} \ . \tag{1}$$

In general, the objective is to find a candidate solution point $x \in S$ in the search space with a low objective function value (w.l.o.g. we consider minimization only). In this work we will additionally confine ourselves to the black-box scenario. The function $f$ is considered as black box and information on $f$ can only be aquired by its evaluation. A sequence of pairs $(x, f(x))$ can be generated and utilized in order to propose further candidate solutions. Evaluations of $f$ are regarded as expensive and therefore the objective becomes two-fold. (1) achieving a solution with a low objective function value and (2) evaluating only a small number of candidate solutions. These two objectives are conflicting.

## 1.1 Historical Ontology

Evolutionary Algorithms (EAs) can be historically divided into four branches.

- Evolutionary Programming [25] invented in the 1960's for evolving Finite State Machines.
- Evolution Strategies (ESs) [65] invented in the 1960's for experimental optimization in engineering.
- Genetic Algorithms (GAs) [41] invented in the 1960's for the studies of adaptive behavior with cellular automata and particularly popularized with [27].
- Genetic Programming [51] invented in the 1990's for evolving computer programs.

The probably most important difference of the approaches relates to their representation of the search problem. *Evolution Strategies* and *Evolutionary Programming* use a real-parameter vector and are considered to mimic evolution on a phenotype level. *Genetic Algorithms* use binary (bit-string) representation and mimic

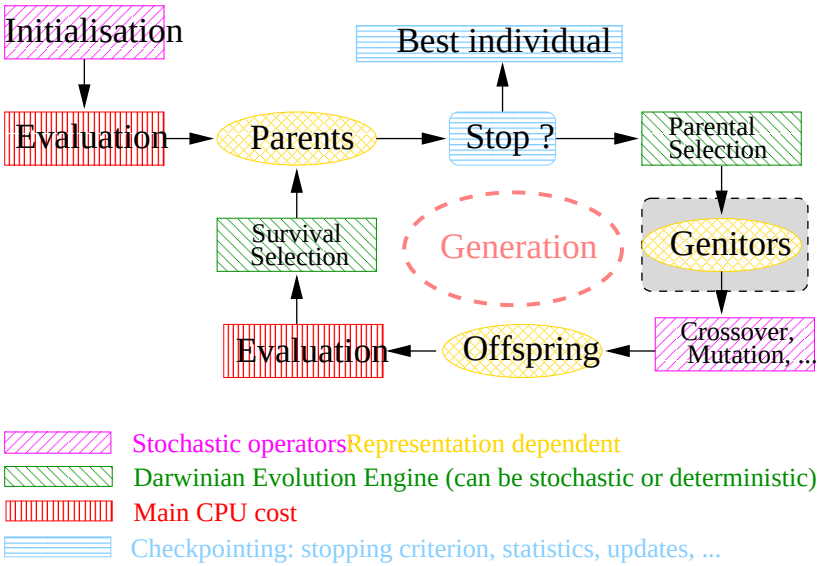


Figure 1: Evolutionary view of randomized black-box search. With kind permission of Marc Schoenauer

a biological genotype. More recently, *real-coded* Genetic Algorithms have been proposed [18, 23] using an entirely different set of mutation and recombination operators than their binary counterparts. *Genetic Programming* operates on tree representations. In the following we focus our discussion to real-parameter representations and investigate EAs as stochastic search procedures.

### 1.2 A Generic Stochastic Search Procedure

In biological evolution *individuals* are a natural entity. Consequently, bio-inspired optimization methods are often viewed and displayed as operating on a population of individuals, that is, in other words, a set of candidate solutions. The population cycle of an evolutionary algorithm is displayed in Figure 1. The main components of the cycle are *parental selection—crossover, mutation—evaluation—survival select*. The sets of Genitors and Offspring (yellow ovals) are completely replaced when operators "Selection" or "Crossover, Mutation,..." are applied respectively (rectangular boxes). The Parents set might or might not be replaced entirely with the Survival Selection. For example, in elitist strategies the best individual will always survive in the Parents population. *New* candidate solutions are only generated

**Initialize** distribution parameters $\boldsymbol{\theta}$
while termination criterion not met
  **Sample** $\lambda$ independent points from distribution $P(\boldsymbol{x}|\boldsymbol{\theta}) \rightarrow \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$
  **Evaluate** the sampled candidate solutions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on $f$
  **Update** parameters $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, (\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_\lambda, f(\boldsymbol{x}_\lambda)))$

Figure 2: Probabilistic view of randomized black-box search. $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function

for the Offspring population.

One important aspect is that systematic improvements, as a rule, only take place with the green Selection operators whereas the Crossover, Mutation,... operator will usually not lead to an improvement of solutions.

Figure 2 takes a different viewpoint of a stochastic search procedure. Here, the procedure is centered around sampling a parameterized distribution. This viewpoint was adopted with *estimation of distribution algorithms* [52]. Only the distribution to generate a new Offspring population is parameterized. This is sufficient, as only the Offspring population actually contains *new* candidate solution points in Figure 2. All remaining operators are integrated in the Update parameters prodecure, where the parameters of the sample distribution are updated. In principle, any stochastic search algorithm can be displayed in both view points. Evolution strategies can often easily be cast into the probabilistic view [49] and the development of *covariance matrix adaptation* [37] was facilitated by this view point.

### 1.3 The Objective Revised

Assuming that $S \subset \mathbb{R}^n$ is a subset of $\mathbb{R}^n$, then a more concise requirement on a search algorithm is to convergence to the optimum in $\overline{S}$. This objective is feasible, given some reasonable assumptions on the properties of $f$. Even more concise we would like to associate a *rate* to the convergence. Evolution strategies indeed have been proved, in particular cases, to exhibit *log-linear convergence* [6], where for the distance to the global optimum, $R_k$, holds

$$\begin{aligned}
\lim_{k\to\infty} \frac{1}{k} \log \frac{R_k}{R_0} &= \lim_{k\to\infty} \frac{1}{k} \left(\log R_k - \log R_0\right) \\
&= \lim_{k\to\infty} \frac{1}{k} \log R_k - \lim_{k\to\infty} \frac{1}{k} \log R_0 \\
&= \lim_{k\to\infty} \frac{1}{k} \log R_k \\
&= \frac{c}{n} \ . \qquad (2)
\end{aligned}$$

Here, $k$ denotes the iteration step and $c$ is a constant. We can conclude that the convergence rate is independent of the initial distance to the optimum $R_0$. For a negative constant $c$ the algorithm converges to the optimum (for $c > 0$ it diverges), where $\exp(c)$ reflects the factor by which the distance to the optimum is reduced in $n$ iteration steps.

### 1.4 Variable Metric

It has been recognized very early during the research on stochastic search algorithms that the parameters of the sample or mutation distribution need to be adaptive. In particular, an appropriate step-size has to be adapted [65, 72]. In a next step the variances in each coordinate have been adapted individually [65, 73], while the random realizations in each coordinate were still independent. The final step was the introduction of *correlated mutations* [74] or a covariance matrix [36, 37][1]. For multivariate normal distributions this means to control all pairwise covariances between variables. The resulting covariance matrix defines, in fact, a Mahalanobis metric in the search space. For the identity as covariance matrix the usual Euclidean metric is recovered. Choosing the covariance matrix for sampling with a multivariate normal distribution resembles variable metric methods like, for example, quasi-Newton methods [1, 16, 28] in gradient based search. The estimate of the inverse Hessian matrix in quasi-Newton methods serves the same purpose as the estimate of the covariance matrix in stochastic search algorithms. In particular for ill-conditioned problems an appropriate estimate can be decisive for the performance.

[1] In [75] an iterative estimate of the Hessian matrix has been introduced into *simultaneous perturbation stochastic approximation* (SPSA). Due to the lack of an adaptive step-size control, this method is still of limited use in practice. *Evolutionary Gradient Search* (EGS) [70] is a method very similar to SPSA but includes step-size control and has been complemented with *covariance matrix adaptation* (CMA) [37] in [5]. In contrast to evolution strategies, SPSA and EGS not only use an ordering of solutions but also their objective function values.

### 1.5 No Free Lunch?

The *No Free Lunch Theorems* (NFL) [63, 83, 80, 42, 82] reveal principle limitations of search or optimization in the discrete domain. The NFL states that taken over *all* functions no algorithm can perform better than another, given that solutions are never revisited. A refinement of the theorem argues that this is true if and only if a set of functions is closed under permutation [43, 80]. If only specific algorithms are considered, the NFL can hold on even smaller sets of functions [82]. In continuous domain NFL is not sustained [8], but it has been argued that computers are necessarily confined to a discrete search space. The NFL suggests that it is impossible to find a generally well-performing algorithm, because, roughly speaking, there always exist as many problems, where this alorithm will perform badly as it will perform well on other ones. Fortunately, NFL is virtually irrelevant in practice, because either the search space under consideration is unrealistically small and even enumeration could be a reasonable option, or the number of functions that need to be considered is way too large to be ever visited even once.

### 1.6 This Thesis

This thesis discusses stochastic variable metric methods for evolution strategies and for continuous domain evolutionary or stochastic optimization algorithms. First, we present the covariance matrix adaptation evolution strategy in depth, which is considered state-of-the-art in continuous domain evolutionary computation [12]. Then we proof that this method can, in principle, be applied to a wide range of search algorithms in continuous domain.

## 2 Covariance Matrix Adaptation Evolution Strategy

This section gives a concise introduction into covariance matrix adaptation and is partly taken from a manuscript written for lecturing third year students.[2]

In the Covariance Matrix Adaptation (CMA) Evolution Strategy newly generated solutions obey a multivariate normal distribution. For given (co-)variances, the normal distribution has the largest entropy of all distributions in $\mathbb{R}^n$. Referring to Fig. 2, the view point adopted in this section, the parameter vector $\boldsymbol{\theta}$ of the distribution consists of a distribution mean, $\boldsymbol{m}$, and a covariance matrix, $\boldsymbol{C}$.

[2] The CMA Evolution Strategy: A Tutorial [http://www.lri.fr/~hansen/cmatutorial.pdf]

## 2.1 Basic equation

In the CMA Evolution Strategy, a population of new search points (individuals, offspring) is generated by sampling a multivariate normal distribution. The basic equation for sampling the search points, for generation number $g = 0, 1, 2, \ldots$, reads

$$\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m}^{(g)} + \sigma^{(g)} \mathcal{N}\left(\mathbf{0}, \boldsymbol{C}^{(g)}\right) \qquad \text{for } k = 1, \ldots, \lambda \tag{3}$$

where

$\sim$ denotes the same distribution on the left and right side.

$\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix $\boldsymbol{C}^{(g)}$. It holds $\boldsymbol{m}^{(g)} + \sigma^{(g)} \mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)}) \sim \mathcal{N}\left(\boldsymbol{m}^{(g)}, (\sigma^{(g)})^2 \boldsymbol{C}^{(g)}\right)$.

$\boldsymbol{x}_k^{(g+1)} \in \mathbb{R}^n$, $k$-th offspring (individual, search point) from generation $g + 1$.

$\boldsymbol{m}^{(g)} \in \mathbb{R}^n$, mean value of the search distribution at generation $g$.

$\sigma^{(g)} \in \mathbb{R}_+$, "overall" standard deviation, step-size, at generation $g$.

$\boldsymbol{C}^{(g)} \in \mathbb{R}^{n \times n}$, covariance matrix at generation $g$. Up to the scalar factor ${\sigma^{(g)}}^2$, $\boldsymbol{C}^{(g)}$ is the covariance matrix of the search distribution.

$\lambda \geq 2$, population size, sample size, number of offspring.

To define a complete iteration step, the remaining question is, how to update the distribution parameters (last line in Fig. 2). Here, we need to calculate $\boldsymbol{m}^{(g+1)}$, $\boldsymbol{C}^{(g+1)}$, and $\sigma^{(g+1)}$ for the next generation $g + 1$. The next three sections will answer these questions in turn.

## 2.2 Selection and Recombination: Moving the Mean

The new mean $\boldsymbol{m}^{(g+1)}$ of the search distribution is a *weighted average of $\mu$ selected points* from the sample $\boldsymbol{x}_1^{(g+1)}, \ldots, \boldsymbol{x}_\lambda^{(g+1)}$:

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_{i:\lambda}^{(g+1)} \tag{4}$$

$$\sum_{i=1}^{\mu} w_i \quad = \quad 1, \qquad w_1 \geq w_2 \geq \cdots \geq w_\mu > 0 \tag{5}$$

where

$\mu \leq \lambda$ is the parent population size, *i.e.* the number of selected points.

$w_{i=1\ldots\mu} \in \mathbb{R}_+$, positive weight coefficients for recombination. For $w_{i=1\ldots\mu} = 1/\mu$, Equation (4) calculates the mean value of $\mu$ selected points.

$\boldsymbol{x}_{i:\lambda}^{(g+1)}$, $i$-th best individual out of $\boldsymbol{x}_1^{(g+1)}, \ldots, \boldsymbol{x}_\lambda^{(g+1)}$ from (3). The index $i : \lambda$ denotes the index of the $i$-th ranked individual and $f(\boldsymbol{x}_{1:\lambda}^{(g+1)}) \leq f(\boldsymbol{x}_{2:\lambda}^{(g+1)}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda}^{(g+1)})$, where $f$ is the objective function to be minimized.

Equation (4) implements *truncation selection* by choosing $\mu < \lambda$ out of $\lambda$ offspring points. Equation (4) implements *weighted intermediate recombination* by taking $\mu > 1$ individuals into account for a weighted average. Assigning *different* weights $w_i$ is an additional selection mechanism and can increase the maximal convergence rate on the sphere function $f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$ by about 25% [3]. *Intermediate* recombination allows for fast convergence rates in combination with a large step-size. The latter is favorable in order to circumvent a failure due to ruggedness of the objective function landscape. The effect of intermediate recombination was interpreted as *genetic repair* in [10].

The measure

$$\mu_{\text{eff}} = \left( \sum_{i=1}^{\mu} w_i^2 \right)^{-1} \tag{6}$$

will be repeatedly used in the following and can be paraphrased as *variance effective selection mass*. From the definition of $w_i$ in (5) we can easily derive $1 \leq \mu_{\text{eff}} \leq \mu$, and $\mu_{\text{eff}} = \mu$ for equal recombination weights, *i.e.* $w_i = 1/\mu$ for all $i = 1 \ldots \mu$. Usually, $\mu_{\text{eff}} \approx \lambda/4$ indicates a reasonable setting of $w_i$. A typical setting could be $w_i \propto \mu - i + 1$, and $\mu \approx \lambda/2$.

### 2.3 Adapting the Covariance Matrix

In this section, the update of the covariance matrix, $\boldsymbol{C}$, is derived. We will start out estimating the covariance matrix from a single population of one generation

(Sect. 2.3.1). For small populations this estimation is unreliable and an adaptation procedure has to be invented (rank-$\mu$-update, Sect. 2.3.1). In the limit case only a single point can be used to update (adapt) the covariance matrix at each generation (rank-one-update, Sect. 2.3.2). The adaptation can be enhanced by exploiting dependencies between successive steps applying cumulation (Sect. 2.3.2). Finally we combine the rank-$\mu$ and rank-one updating methods (Sect. 2.3.3).

### 2.3.1 Rank-$\mu$ Update

**Estimating the Covariance Matrix From Scratch** For the moment we assume that the population contains enough information to reliably estimate a covariance matrix from the population.[3] For the sake of convenience we assume $\sigma^{(g)} = 1$ (see (3)) in this section. For $\sigma^{(g)} \neq 1$ the formulae hold except for a constant factor.

We can (re-)estimate the original covariance matrix $\boldsymbol{C}^{(g)}$ using the sampled population from (3), $\boldsymbol{x}_1^{(g+1)} \ldots \boldsymbol{x}_\lambda^{(g+1)}$, via the empirical covariance matrix

$$\boldsymbol{C}_{\text{emp}}^{(g+1)} = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left( \boldsymbol{x}_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \boldsymbol{x}_j^{(g+1)} \right) \left( \boldsymbol{x}_i^{(g+1)} - \frac{1}{\lambda} \sum_{j=1}^{\lambda} \boldsymbol{x}_j^{(g+1)} \right)^{\mathrm{T}} . \tag{7}$$

The empirical covariance matrix $\boldsymbol{C}_{\text{emp}}^{(g+1)}$ is an unbiased estimator of $\boldsymbol{C}^{(g)}$: assuming the $\boldsymbol{x}_i^{(g+1)}, i = 1 \ldots \lambda$, to be random variables (rather than a realized sample), we have that $\mathsf{E}\left[\boldsymbol{C}_{\text{emp}}^{(g+1)} \,\middle|\, \boldsymbol{C}^{(g)}\right] = \boldsymbol{C}^{(g)}$. Consider now a slightly different approach to get an estimator for $\boldsymbol{C}^{(g)}$.

$$\boldsymbol{C}_{\lambda}^{(g+1)} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left( \boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)} \right) \left( \boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)} \right)^{\mathrm{T}} \tag{8}$$

Also the matrix $\boldsymbol{C}_{\lambda}^{(g+1)}$ is an unbiased estimator of $\boldsymbol{C}^{(g)}$. The remarkable difference between (7) and (8) is the reference mean value. For $\boldsymbol{C}_{\text{emp}}^{(g+1)}$ it is the mean of the *actually realized* sample. For $\boldsymbol{C}_{\lambda}^{(g+1)}$ it is the *true* mean value, $\boldsymbol{m}^{(g)}$, of the sampled distribution (see (3)). Therefore, the estimators $\boldsymbol{C}_{\text{emp}}^{(g+1)}$ and $\boldsymbol{C}_{\lambda}^{(g+1)}$ can be interpreted differently: while $\boldsymbol{C}_{\text{emp}}^{(g+1)}$ estimates the distribution variance *within the sampled points*, $\boldsymbol{C}_{\lambda}^{(g+1)}$ estimates variances of sampled *steps*, $\boldsymbol{x}_i^{(g+1)} - \boldsymbol{m}^{(g)}$.

[3] In order to re-estimate the covariance matrix, $\boldsymbol{C}$, out of $\lambda\, \mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed samples such that $\mathtt{cond}(\boldsymbol{C}) < 10$, a sample size $\lambda \geq 4n$ is needed, as can be easily verified in a numerical experiment.

A minor difference between (7) and (8) is the different normalizations $\frac{1}{\lambda-1}$ versus $\frac{1}{\lambda}$, necessary to get an unbiased estimator in both cases. In (7) one degree of freedom is already taken by the inner summand. In order to get a maximum likelihood estimator in both cases $\frac{1}{\lambda}$ must be used.

Equation (8) re-estimates *the original* covariance matrix. To "estimate" a "better" covariance matrix (8) is modified and the same, *weighted selection* mechanism as in (4) is used [33].

$$\boldsymbol{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left( \boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)} \right) \left( \boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)} \right)^{\mathrm{T}} \tag{9}$$

The matrix $\boldsymbol{C}_{\mu}^{(g+1)}$is an estimator for the distribution of *selected steps*, just as $\boldsymbol{C}_{\lambda}^{(g+1)}$ is an estimator of the original distribution of steps before selection. Sampling from $\boldsymbol{C}_{\mu}^{(g+1)}$ tends to reproduce selected, *i.e. successful* steps, giving a justification for what a "better" covariance matrix means.

Following [30], we compare (9) with the Estimation of Multivariate Normal Algorithm EMNA$_{global}$ [53]. The covariance matrix in EMNA$_{global}$ reads, analogously to (7),

$$\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \left( \boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)} \right) \left( \boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g+1)} \right)^{\mathrm{T}} \quad , \tag{10}$$

where $\boldsymbol{m}^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{x}_{i:\lambda}^{(g+1)}$. Similarly, applying the so-called Cross-Entropy method to continuous domain optimization [67] yields the covariance matrix $\frac{\mu}{\mu-1} \boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$, *i.e.* the *unbiased* empirical covariance matrix of the $\mu$ best points. In both cases the subtle, but most important difference to (9) is, again, the choice of the reference mean value.[4] Equation (10) estimates the variance *within* the selected population while (9) estimates selected steps. Equation (10) reveals always smaller variances than (9), because its reference mean value is the minimizer for the variances. Moreover, in most conceivable selection situations (10) decreases the variances compared to $\boldsymbol{C}^{(g)}$.

Figure 3 demonstrates the estimation results on *a linear* objective function for $\lambda = 150$, $\mu = 50$, and $w_i = 1/\mu$. Equation (9) increases the expected variance in direction of the gradient (where the selection takes place, here the diagonal), given ordinary settings for parent number $\mu$ and recombination weights $w_1, \ldots, w_\mu$. Equation (10) always decreases the variance

[4] Taking a weighted sum, $\sum_{i=1}^{\mu} w_i \ldots$, instead of the mean, $\frac{1}{\mu} \sum_{i=1}^{\mu} \ldots$, is an appealing, but less important, difference.

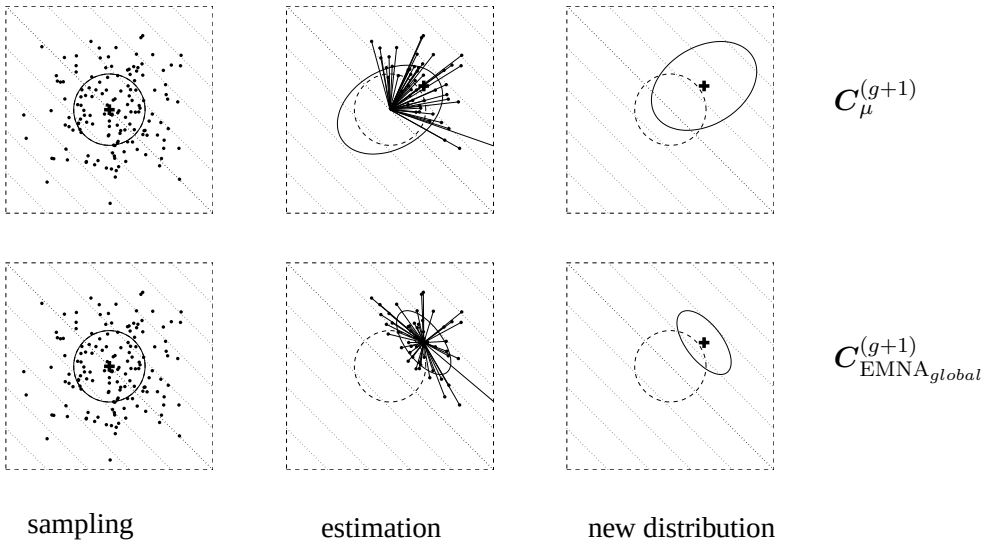


Figure 3: Estimation of the covariance matrix on $f_{\mathrm{linear}}(\boldsymbol{x}) = -\sum_{i=1}^{2} x_i$ to be minimized. Contour lines (*dotted*) indicate that the strategy should move toward the upper right corner. **Above**: estimation of $\boldsymbol{C}_\mu^{(g+1)}$ according to (9), where $w_i = 1/\mu$. **Below**: estimation of $\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$ according to (10). Left: sample of $\lambda = 150\,\mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed points. Middle: the $\mu = 50$ selected points (*dots*) determining the entries for the estimation equation (*solid straight lines*). Right: search distribution of the next generation (*solid ellipsoids*). Given $w_i = 1/\mu$, estimation via $\boldsymbol{C}_\mu^{(g+1)}$ *increases* the expected variance in gradient direction for all $\mu < \lambda/2$, while estimation via $\boldsymbol{C}_{\mathrm{EMNA}_{global}}^{(g+1)}$ *decreases* this variance for any $\mu < \lambda$

> in gradient direction! Therefore, (10) is highly susceptible to premature convergence, in particular with small parent populations, where the population cannot be expected to bracket the optimum at any time. However, for large values of $\mu$ in large populations with large initial variances, the impact of the different reference mean value can become marginal.

In order to ensure with (3), (4), and (9), that $\boldsymbol{C}_\mu^{(g+1)}$ is a *reliable* estimator, the variance effective selection mass $\mu_{\mathrm{eff}}$ (cf. (6)) must be large enough: getting condition numbers smaller than ten for $\boldsymbol{C}_\mu^{(g)}$ on $f_{\mathrm{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, requires $\mu_{\mathrm{eff}} \approx 10n$. The next step is to circumvent this restriction on $\mu_{\mathrm{eff}}$.

**Rank-$\mu$-Update** To achieve *fast* search (opposite to *more robust* or *more global* search), e.g. competitive performance on $f_{\text{sphere}}$, the population size $\lambda$ must be small. Because $\mu_{\text{eff}} \approx \lambda/4$ also $\mu_{\text{eff}}$ must be small and we may assume, e.g., $\mu_{\text{eff}} \leq 1 + \ln n$. Then, it is not possible to get a *reliable* estimator for a good covariance matrix from (9). As a remedy, information from previous generations is used additionally. For example, after a sufficient number of generations, the mean of the estimated covariance matrices from all generations,

$$\boldsymbol{C}^{(g+1)} = \frac{1}{g+1} \sum_{i=0}^{g} \frac{1}{{\sigma^{(i)}}^2} \boldsymbol{C}_{\mu}^{(i+1)} \tag{11}$$

becomes a reliable estimator for the selected steps. To make $\boldsymbol{C}_{\mu}^{(g)}$ from different generations comparable, the different $\sigma^{(i)}$ are incorporated. (We also assume that the covariance matrices $\boldsymbol{c}_{\mu}$ do not decrease fast, because otherwise old information would dominate (11). Assuming $\sigma^{(i)} = 1$, (11) resembles the covariance matrix from the Estimation of Multivariate Normal Algorithm EMNA$_i$ [53].)

In (11), all generation steps have the same weight. To assign recent generations a higher weight, exponential smoothing is introduced. Choosing $\boldsymbol{C}^{(0)} = \mathbf{I}$ to be the unity matrix and a learning rate $0 < c_{\text{cov}} \leq 1$, then $\boldsymbol{C}^{(g+1)}$ reads

$$\begin{aligned} \boldsymbol{C}^{(g+1)} &= (1 - c_{\text{cov}}) \boldsymbol{C}^{(g)} + c_{\text{cov}} \frac{1}{{\sigma^{(g)}}^2} \boldsymbol{C}_{\mu}^{(g+1)} \\ &= (1 - c_{\text{cov}}) \boldsymbol{C}^{(g)} + c_{\text{cov}} \sum_{i=1}^{\mu} w_i \, \boldsymbol{y}_{i:\lambda}^{(g+1)} {\boldsymbol{y}_{i:\lambda}^{(g+1)}}^{\mathrm{T}} \end{aligned} \tag{12}$$

where

$c_{\text{cov}} \leq 1$ learning rate for updating the covariance matrix. For $c_{\text{cov}} = 1$, no prior information is retained and $\boldsymbol{C}^{(g+1)} = \frac{1}{{\sigma^{(g)}}^2} \boldsymbol{C}_{\mu}^{(g+1)}$. For $c_{\text{cov}} = 0$, no learning takes place and $\boldsymbol{C}^{(g+1)} = \boldsymbol{C}^{(0)}$. Here, $c_{\text{cov}} \approx \min(1, \mu_{\text{eff}}/n^2)$ is a reasonably choice.

$\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

This covariance matrix update is called rank-$\mu$-update [34], because the sum of outer products in (12) is of rank $\min(\mu, n)$ (with probability one). Note that this sum can even consist of a single term, if $\mu = 1$.

The number $1/c_{\text{cov}}$ is the *backward time horizon* that contains roughly $63\%$ of the overall weight.

Because (12) expands to the weighted sum

$$\boldsymbol{C}^{(g+1)} = (1-c_{\text{cov}})^{g+1}\boldsymbol{C}^{(0)} + c_{\text{cov}} \sum_{i=0}^{g} (1-c_{\text{cov}})^{g-i} \frac{1}{{\sigma^{(i)}}^2} \boldsymbol{C}_{\mu}^{(i+1)} \ , \quad (13)$$

the backward time horizon, $\Delta g$, where about $63\%$ of the overall weight is summed up, is defined by

$$c_{\text{cov}} \sum_{i=g+1-\Delta g}^{g} (1-c_{\text{cov}})^{g-i} \approx 0.63 \approx 1 - \frac{1}{\mathrm{e}} \ . \quad (14)$$

Resolving the sum yields

$$(1-c_{\text{cov}})^{\Delta g} \approx \frac{1}{\mathrm{e}} \ , \quad (15)$$

and resolving for $\Delta g$, using the Taylor approximation for $\ln$, yields

$$\Delta g \approx \frac{1}{c_{\text{cov}}} \ . \quad (16)$$

That is, approximately $37\%$ of the information in $\boldsymbol{C}^{(g+1)}$ is older than $1/c_{\text{cov}}$ generations, and, according to (15), the original weight is reduced by a factor of $0.37$ after approximately $1/c_{\text{cov}}$ generations.[5]

The choice of $c_{\text{cov}}$ is crucial. Small values lead to slow learning, too large values lead to a failure, because the covariance matrix degenerates. Fortunately, a good setting seems to be largely independent of the function to be optimized.[6] A first order approximation for a good choice is $c_{\text{cov}} \approx \mu_{\text{eff}}/n^2$. Therefore, the characteristic time horizon for (12) is roughly $n^2/\mu_{\text{eff}}$.

Even for the learning rate $c_{\text{cov}} = 1$, adapting the covariance matrix cannot be accomplished within one generation. The effect of the original sample distribution does not vanish until a sufficient number of generations. Assuming fixed search costs (number of function evaluations), a small population size $\lambda$ allows a larger number of generations and therefore usually leads to a faster adaptation of the covariance matrix.

[5] This can be shown more easily, because $(1-c_{\text{cov}})^g = \exp \ln (1-c_{\text{cov}})^g = \exp(g \ln(1-c_{\text{cov}})) \approx \exp(-g c_{\text{cov}})$ for small $c_{\text{cov}}$, and for $g \approx 1/c_{\text{cov}}$ we get immediately $(1-c_{\text{cov}})^g \approx \exp(-1)$.

[6] We use the sphere model $f_{\text{sphere}}(\boldsymbol{x}) = \sum_i x_i^2$ to empirically find a good setting for the parameter $c_{\text{cov}}$, dependent on $n$ and $\mu_{\text{eff}}$, realizing a condition number of about three. The realized condition number is monotonous in $c_{\text{cov}}$. The found setting was reasonably applicable to any non-noisy objective function we have tried so far.

### 2.3.2 Rank-One-Update

In Section 2.3.1 we estimated the complete covariance matrix from scratch, using all selected steps from a *single generation*. We now take precisely the opposite viewpoint. We will repeatedly *update* the covariance matrix in the generation sequence using a *single selected step* only. First, this perspective will give a justification of the adaptation rule (12). Second, we will introduce the so-called evolution path that is finally used for a rank-one update of the covariance matrix.

**A Different Viewpoint** We consider a specific method to produce $n$-dimensional normal distributions with zero mean. Let the vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{g_0} \in \mathbb{R}^n$, $g_0 \geq n$, span $\mathbb{R}^n$ and let $\mathcal{N}(0,1)$ denote independent $(0,1)$-normally distributed random numbers, then

$$\mathcal{N}(0,1)\,\boldsymbol{y}_1 + \cdots + \mathcal{N}(0,1)\,\boldsymbol{y}_{g_0} \quad \sim \quad \mathcal{N}\left(\boldsymbol{0}, \sum_{i=1}^{g_0} \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}}\right) \tag{17}$$

is a normally distributed random vector with zero mean and covariance matrix $\sum_{i=1}^{g_0} \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}}$. The random vector (17) is generated by adding "line-distributions" $\mathcal{N}(0,1)\,\boldsymbol{y}_i$. The singular distribution $\mathcal{N}(0,1)\,\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{y}_i \boldsymbol{y}_i^{\mathrm{T}})$ generates the vector $\boldsymbol{y}_i$ with maximum likelihood considering all normal distributions with zero mean.

> The line distribution that generates a vector $\boldsymbol{y}$ with the maximum likelihood must "live" on a line that includes $\boldsymbol{y}$, and therefore the distribution must obey $\mathcal{N}(0,1)\sigma\boldsymbol{y} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{y}\boldsymbol{y}^{\mathrm{T}})$. Any other line distribution with zero mean cannot generate $\boldsymbol{y}$ at all. Choosing $\sigma$ reduces to choosing the maximum likelihood of $\|\boldsymbol{y}\|$ for the one-dimensional gaussian $\mathcal{N}(0, \sigma^2 \|\boldsymbol{y}\|^2)$, which is $\sigma = 1$.
>
> The covariance matrix $\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}$ has rank one, its only eigenvectors are $\mathbb{R}_{\setminus 0} \times \boldsymbol{y}$ with eigenvalue $\|\boldsymbol{y}\|^2$. Using equation (17), any normal distribution can be realized if $\boldsymbol{y}_i$ are chosen appropriately. In general, the vectors $\boldsymbol{y}_i$ need not to be eigenvectors of the covariance matrix (and usually are not).

Considering (17) and a slight simplification of (12), we try to gain insight into the adaptation rule for the covariance matrix. Let the sum in (12) consist of a single summand only (*e.g.* $\mu = 1$), and let $\boldsymbol{y}_{g+1} = \frac{\boldsymbol{x}_{1:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}$. Then, the rank-one update for the covariance matrix reads

$$\boldsymbol{C}^{(g+1)} \quad = \quad (1 - c_{\mathrm{cov}})\boldsymbol{C}^{(g)} + c_{\mathrm{cov}}\, \boldsymbol{y}_{g+1} {\boldsymbol{y}_{g+1}}^{\mathrm{T}} \tag{18}$$

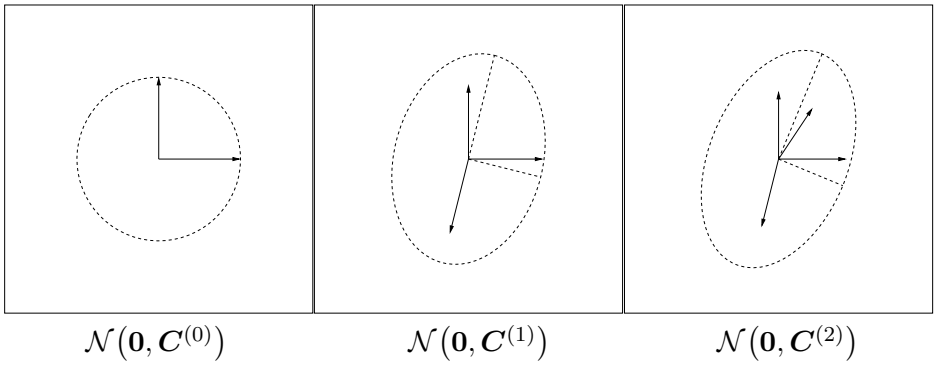


Figure 4: Change of the distribution according to the covariance matrix update (18). Left: vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, and $\boldsymbol{C}^{(0)} = \mathbf{I} = \boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} + \boldsymbol{e}_2\boldsymbol{e}_2^{\mathrm{T}}$. Middle: vectors $0.91\,\boldsymbol{e}_1$, $0.91\,\boldsymbol{e}_2$, and $0.41\,\boldsymbol{y}_1$ (the coefficients deduce from $c_{\mathrm{cov}} = 0.17$), and $\boldsymbol{C}^{(1)} = (1 - c_{\mathrm{cov}})\,\mathbf{I} + c_{\mathrm{cov}}\,\boldsymbol{y}_1\boldsymbol{y}_1^{\mathrm{T}}$, where $\boldsymbol{y}_1 = \binom{-0.59}{-2.2}$. The distribution ellipsoid is elongated into the direction of $\boldsymbol{y}_1$, and therefore increases the likelihood of $\boldsymbol{y}_1$. Right: $\boldsymbol{C}^{(2)} = (1 - c_{\mathrm{cov}})\,\boldsymbol{C}^{(1)} + c_{\mathrm{cov}}\,\boldsymbol{y}_2\boldsymbol{y}_2^{\mathrm{T}}$, where $\boldsymbol{y}_2 = \binom{0.97}{1.5}$.

The right summand is of rank one and adds the maximum likelihood term for $\boldsymbol{y}_{g+1}$ into the covariance matrix $\boldsymbol{C}^{(g)}$. Therefore the probability to generate $\boldsymbol{y}_{g+1}$ in the next generation increases.

An example of the first two iteration steps of (18) is shown in **Figure 4**. The distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(1)})$ tends to reproduce $\boldsymbol{y}_1$ with a larger probability than the initial distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$; the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(2)})$ tends to reproduce $\boldsymbol{y}_2$ with a larger probability than $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(1)})$, and so forth. When $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_g$ denote the formerly selected, favorable steps, $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ tends to reproduce these steps. The process leads to an alignment of the search distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$ to the distribution of the selected steps. If both distributions become alike, as under random selection, in expectation no further change of the covariance matrix takes place [29].

**Cumulation: Utilizing the Evolution Path** We have used the selected steps, $\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$, to update the covariance matrix in (12) and (18). Because $\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}} = -\boldsymbol{y}(-\boldsymbol{y})^{\mathrm{T}}$, *the sign of the steps is irrelevant* for the update of the covariance matrix — that is, the sign information is not used for calculating $\boldsymbol{C}^{(g+1)}$. To exploit sign information, the so-called *evolution path* is introduced [36, 37].

We call a sequence of successive steps, the strategy takes over a number of

generations, an evolution path. An evolution path can be expressed by a sum of consecutive steps. This summation is referred to as *cumulation*. To construct an evolution path, the step-size $\sigma$ is disregarded. For example, an evolution path of three steps of the distribution mean $\boldsymbol{m}$ can be constructed by the sum

$$\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} + \frac{\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}}{\sigma^{(g-1)}} + \frac{\boldsymbol{m}^{(g-1)} - \boldsymbol{m}^{(g-2)}}{\sigma^{(g-2)}} \quad . \tag{19}$$

In practice, to construct the evolution path, $\boldsymbol{p}_\mathrm{c} \in \mathbb{R}^n$, we use exponential smoothing as in (12), and start with $\boldsymbol{p}_\mathrm{c}^{(0)} = \mathbf{0}$.[7]

$$\boldsymbol{p}_\mathrm{c}^{(g+1)} = (1 - c_\mathrm{c})\boldsymbol{p}_\mathrm{c}^{(g)} + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{eff}} \, \frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \tag{20}$$

where

$\boldsymbol{p}_\mathrm{c}^{(g)} \in \mathbb{R}^n$, evolution path at generation $g$.

$c_\mathrm{c} \leq 1$. Again, $1/c_\mathrm{c}$ is the backward time horizon of the evolution path $\boldsymbol{p}_\mathrm{c}$ that contains roughly $63\%$ of the overall weight (compare derivation of (16)). A time horizon between $\sqrt{n}$ and $n$ is reasonable [29].

The factor $\sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{eff}}$ is a normalization constant for $\boldsymbol{p}_\mathrm{c}$. For $c_\mathrm{c} = 1$ and $\mu_\mathrm{eff} = 1$, the factor reduces to one, and $\boldsymbol{p}_\mathrm{c}^{(g+1)} = (\boldsymbol{x}_{1:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

The factor $\sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{eff}}$ is chosen, such that

$$\boldsymbol{p}_\mathrm{c}^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \tag{21}$$

if

$$\boldsymbol{p}_\mathrm{c}^{(g)} \sim \frac{\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \quad \text{for all } i = 1, \ldots, \mu \quad . \tag{22}$$

To derive (21) from (22) and (20) remark that

$$(1-c_\mathrm{c})^2 + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c})}^2 = 1 \quad \text{and} \quad \sum_{i=1}^{\mu} w_i \mathcal{N}_i(\mathbf{0}, \boldsymbol{C}) \sim \frac{1}{\sqrt{\mu_\mathrm{eff}}} \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \quad . \tag{23}$$

[7] In the final algorithm (20) is still slightly modified.

The (rank-one) update of the covariance matrix $\boldsymbol{C}^{(g)}$ via the evolution path $\boldsymbol{p}_{\mathrm{c}}^{(g+1)}$ reads [36]

$$\boldsymbol{C}^{(g+1)} = (1 - c_{\mathrm{cov}})\boldsymbol{C}^{(g)} + c_{\mathrm{cov}} \boldsymbol{p}_{\mathrm{c}}^{(g+1)} \boldsymbol{p}_{\mathrm{c}}^{(g+1)\,\mathrm{T}} \quad . \tag{24}$$

An empirically validated choice for the learning rate in (24) is $c_{\mathrm{cov}} \approx 2/n^2$. For $c_{\mathrm{c}} = 1$ and $\mu = 1$, Equations (24), (18), and (12) are identical.

Using the evolution path for the update of $\boldsymbol{C}$ is a significant improvement of (12) for small $\mu_{\mathrm{eff}}$, because correlations between consecutive steps are exploited. The leading signs of steps, and the dependencies between consecutive steps play a significant role for the resulting evolution path $\boldsymbol{p}_{\mathrm{c}}^{(g+1)}$. For $c_{\mathrm{c}} \approx 3/n$ the number of function evaluations needed to adapt a nearly optimal covariance matrix on cigar-like objective functions becomes $\mathcal{O}(n)$.

As a last step, we combine (12) and (24).

### 2.3.3 Combining Rank-$\mu$-Update and Cumulation

The final CMA update of the covariance matrix combines (12) and (24), where $\mu_{\mathrm{cov}}$ determines their relative weighting.

$$\begin{aligned} \boldsymbol{C}^{(g+1)} \quad = \quad & (1 - c_1 - c_\mu)\, \boldsymbol{C}^{(g)} + c_1 \underbrace{\boldsymbol{p}_{\mathrm{c}}^{(g+1)} \boldsymbol{p}_{\mathrm{c}}^{(g+1)\,\mathrm{T}}}_{\text{rank-one update}} \\ & + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda}^{(g+1)} \left(\boldsymbol{y}_{i:\lambda}^{(g+1)}\right)^{\mathrm{T}}}_{\text{rank-}\mu\text{ update}} \end{aligned} \tag{25}$$

where

$0 \leq c_1, c_\mu \leq 1$ and $c_1 + c_\mu \leq 1$

$\boldsymbol{y}_{i:\lambda}^{(g+1)} = (\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}$.

Equation (25) reduces to (12) for $c_1 = 0$ and $c_\mu = c_{\mathrm{cov}}$ and to (24) for $c_\mu = 0$ and $c_1 = c_{\mathrm{cov}}$. The equation combines the advantages of (12) and (24). On the one hand, the information within the population of one generation is used efficiently by the rank-$\mu$ update. On the other hand, information of correlations between generations is exploited by using the evolution path for the rank-one update. The

former is important in large populations, the latter is in particular important in small populations.

### 2.4 Step-Size Control

The covariance matrix adaptation, introduced in the last section, does not explicitly control the "overall scale" of the distribution, the step-size. The covariance matrix adaptation increases the scale only in one direction for each selected step, and it decreases the scale only implicitly by fading out old information via the factor $1 - c_{\text{cov}}$. Less informally, we can state two specific reasons to introduce a step-size control in addition to the adaptation rule (25) for $\boldsymbol{C}^{(g)}$.

1. The optimal overall step length cannot be well approximated by (25), in particular if $\mu_{\text{eff}}$ is chosen larger than one.

   > For example, on $f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, the optimal step-size $\sigma$ equals approximately $\mu \sqrt{f_{\text{sphere}}(\boldsymbol{x})}/n$, given $\boldsymbol{C}^{(g)} \approx \mathbf{I}$ and $\mu_{\text{eff}} = \mu \ll n$ [11, 64]. This dependency on $\mu$ cannot be realized by (12), and is also not well approximated by (25).

2. The largest reliable learning rate for the covariance matrix update in (25) is too slow to achieve competitive change rates for the overall step length.

   > To achieve optimal performance on $f_{\text{sphere}}$ with an Evolution Strategy with equal recombination weights, the overall step length must decrease by a factor of approximately $\exp(0.202) \approx 1.22$ within $n$ function evaluations, as can be derived from progress formulas [11, p. 229]. That is, the time horizon for the step length change must be proportional to $n$ or shorter. From the learning rate $c_{\text{cov}}$ in (25) follows that the adaptation is too slow to perform competitive on $f_{\text{sphere}}$ whenever $\mu_{\text{eff}} \ll n$. This can be validated by simulations even for moderate dimensions, say, $n \geq 10$ and small $\mu_{\text{eff}}$, say, $\mu_{\text{eff}} \leq 1 + \ln n$.

To control the step-size $\sigma^{(g)}$ we utilize an evolution path, *i.e.* a sum of successive steps (see page 14). The method can be applied independently of the covariance matrix update and is denoted as *cumulative path length control*, cumulative step-size control, or *cumulative step length adaptation (CSA)*. The length of an evolution path is exploited, based on the following reasoning (compare also Fig. 5).

- Whenever the evolution path is short, single steps cancel each other out (Fig. 5, left). Loosely speaking, they are anti-correlated. If steps annihilate each other, the step-size should be decreased.

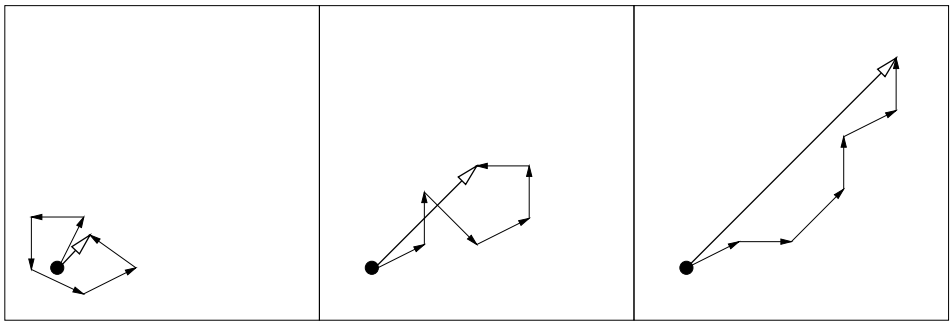

Figure 5: Three evolution paths of respectively six steps from different selection situations (idealized). The lengths of the *single* steps are all comparable. The length of the evolution paths (sum of steps) is remarkably different and is exploited for step-size control

- Whenever the evolution path is long, the single steps are pointing to similar directions (Fig. 5, right). Loosely speaking, they are correlated. Because the steps are similar, the same distance can be covered by fewer but longer steps into the same directions. In the limit case, where consecutive steps have identical direction, they can be replaced by an enlarged single step. Consequently, the step-size should be increased.

- In the desired situation the steps are (approximately) perpendicular in expectation and therefore uncorrelated (Fig. 5, middle).

To decide whether the evolution path is "long" or "short", we compare the length of the path with its *expected length under random selection*.[8] Under random selection consecutive steps are independent and therefore uncorrelated (we just realized that "uncorrelated" steps are the desired situation). If selection biases the evolution path to be longer then expected, $\sigma$ is increased, and, vice versa, if selection biases the evolution path to be shorter than expected, $\sigma$ is decreased. In the ideal situation, selection does not bias the length of the evolution path and the length equals its expected length under random selection.

In practice, to construct the evolution path, $\boldsymbol{p}_\sigma$, the same techniques as in (20) are applied. In contrast to (20), a *conjugate* evolution path is constructed, because

[8] Random selection means that the index $i : \lambda$ (compare (4)) is independent of the value of $\boldsymbol{x}_{i:\lambda}^{(g+1)}$ for all $i = 1, \ldots, \lambda$, e.g. $i : \lambda = i$.

the expected length of the evolution path $\boldsymbol{p}_\mathrm{c}$ from (20) depends on its direction (compare (21)). Initialized with $\boldsymbol{p}_\sigma^{(0)} = \mathbf{0}$, the conjugate evolution path reads

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_\mathrm{eff}}\, \boldsymbol{C}^{(g)^{-\frac{1}{2}}} \frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}} \quad (26)$$

where

$\boldsymbol{p}_\sigma^{(g)} \in \mathbb{R}^n$ is the conjugate evolution path at generation $g$.

$c_\sigma < 1$. Again, $1/c_\sigma$ is the backward time horizon of the evolution path (compare (16)). For small $\mu_\mathrm{eff}$, a time horizon between $\sqrt{n}$ and $n$ is reasonable.

$\sqrt{c_\sigma(2-c_\sigma)\mu_\mathrm{eff}}$ is a normalization constant, see (20).

$\boldsymbol{C}^{(g)^{-\frac{1}{2}}} \stackrel{\mathrm{def}}{=} \boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)^{-1}}\boldsymbol{B}^{(g)^\mathrm{T}}$, where $\boldsymbol{C}^{(g)} = \boldsymbol{B}^{(g)}\left(\boldsymbol{D}^{(g)}\right)^2\boldsymbol{B}^{(g)^\mathrm{T}}$ is an eigendecomposition of $\boldsymbol{C}^{(g)}$, where $\boldsymbol{B}^{(g)}$ is an orthonormal basis of eigenvectors, and the diagonal elements of the diagonal matrix $\boldsymbol{D}^{(g)}$ are square roots of the corresponding positive eigenvalues.

For $\boldsymbol{C}^{(g)} = \mathbf{I}$, we have $\boldsymbol{C}^{(g)^{-\frac{1}{2}}} = \mathbf{I}$ and (26) replicates (20). The transformation $\boldsymbol{C}^{(g)^{-\frac{1}{2}}}$ re-scales the step $\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}$ within the coordinate system given by $\boldsymbol{B}^{(g)}$.

The single factors of the transformation $\boldsymbol{C}^{(g)^{-\frac{1}{2}}} = \boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)^{-1}}\boldsymbol{B}^{(g)^\mathrm{T}}$ can be explained as follows (from right to left):

$\boldsymbol{B}^{(g)^\mathrm{T}}$ rotates the space such that the columns of $\boldsymbol{B}^{(g)}$, *i.e.* the principle axes of the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{C}^{(g)})$, rotate into the coordinate axes. Elements of the resulting vector relate to projections onto the corresponding eigenvectors.

$\boldsymbol{D}^{(g)^{-1}}$ applies a (re-)scaling such that all axes become equally sized.

$\boldsymbol{B}^{(g)}$ rotates the result back into the original coordinate system. This last transformation ensures that the principal axes of the distribution are not rotated by the overall transformation and directions of consecutive steps are comparable.

Consequently, the transformation $\boldsymbol{C}^{(g)^{-\frac{1}{2}}}$ makes the expected length of $\boldsymbol{p}_\sigma^{(g+1)}$ independent of its direction, and for any sequence of realized covariance matrices $\boldsymbol{C}^{(g)}_{g=0,1,2,\ldots}$ we have under random selection $\boldsymbol{p}_\sigma^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, given $\boldsymbol{p}_\sigma^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [29].

To update $\sigma^{(g)}$, we "compare" $\|\boldsymbol{p}_\sigma^{(g+1)}\|$ with its expected length $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$, that is

$$\ln \sigma^{(g+1)} = \ln \sigma^{(g)} + \frac{c_\sigma}{d_\sigma} \left( \frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1 \right) \quad , \tag{27}$$

where

- $d_\sigma \approx 1$, damping parameter, scales the change magnitude of $\ln \sigma^{(g)}$. The factor $c_\sigma / d_\sigma$ is based on in-depth investigations of the algorithm [29].
- $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| = \sqrt{2}\,\Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2}) \approx \sqrt{n}\,(1 - \frac{1}{4n} + \frac{1}{21n^2})$, expectation of the Euclidean norm of a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ distributed random vector.

For $\|\boldsymbol{p}_\sigma^{(g+1)}\| = \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$ the second summand in (27) is zero, and $\sigma^{(g)}$ is unchanged, while $\sigma^{(g)}$ is increased for $\|\boldsymbol{p}_\sigma^{(g+1)}\| > \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$, and $\sigma^{(g)}$ is decreased for $\|\boldsymbol{p}_\sigma^{(g+1)}\| < \mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$.

> Alternatively, we might use the squared norm $\|\boldsymbol{p}_\sigma^{(g+1)}\|^2$ in (27) and compare with its expected value $n$ [4]. In this case (27) would read
>
> $$\ln \sigma^{(g+1)} = \ln \sigma^{(g)} + \frac{c_\sigma}{2d_\sigma} \left( \frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|^2}{n} - 1 \right) \quad . \tag{28}$$
>
> This update will presumable lead to faster step-size increments and slower step-size decrements.

The step-size change is unbiased on the log scale, because $\mathsf{E}\big[\ln \sigma^{(g+1)} \,\big|\, \sigma^{(g)}\big] = \ln \sigma^{(g)}$ for $\boldsymbol{p}_\sigma^{(g+1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Equations (26) and (27) cause successive steps of the distribution mean $\boldsymbol{m}^{(g)}$ to be approximately $\boldsymbol{C}^{(g)^{-1}}$-conjugate.

> In order to show that successive steps are approximately $\boldsymbol{C}^{(g)^{-1}}$-conjugate first we remark that (26) and (27) adapt $\sigma$ such that the length of $\boldsymbol{p}_\sigma^{(g+1)}$ equals approximately $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$. Starting from $(\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|)^2 \approx \|\boldsymbol{p}_\sigma^{(g+1)}\|^2 = \boldsymbol{p}_\sigma^{(g+1)^{\mathrm{T}}} \boldsymbol{p}_\sigma^{(g+1)} = \mathrm{RHS}^{\mathrm{T}}\mathrm{RHS}$ of (26) and assuming that the

expected squared *length* of $\boldsymbol{C}^{(g)^{-\frac{1}{2}}}(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)})$ is unchanged by selection (unlike its direction) we get after some computations

$$\boldsymbol{p}_\sigma^{(g)^{\mathrm{T}}} \boldsymbol{C}^{(g)^{-\frac{1}{2}}}(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}) \approx 0 \ , \tag{29}$$

and

$$\left(\boldsymbol{C}^{(g)^{\frac{1}{2}}} \boldsymbol{p}_\sigma^{(g)}\right)^{\mathrm{T}} \boldsymbol{C}^{(g)^{-1}} \left(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}\right) \approx 0 \ . \tag{30}$$

Assuming that the re-scaled constructing steps from (26) become roughly perpendicular, as under random selection, means

$$\left(\boldsymbol{C}^{(g_1)^{-\frac{1}{2}}} \left(\boldsymbol{m}^{(g_1+1)} - \boldsymbol{m}^{(g_1)}\right)\right)^{\mathrm{T}} \boldsymbol{C}^{(g_2)^{-\frac{1}{2}}} \left(\boldsymbol{m}^{(g_2+1)} - \boldsymbol{m}^{(g_2)}\right) \approx 0 \tag{31}$$

for $g_1 \neq g_2$. For $|g_1 - g_2| \ll 1/c_{\mathrm{cov}}$ we can assume $\boldsymbol{C}^{(g_1)} \approx \boldsymbol{C}^{(g_2)}$, and setting $\boldsymbol{C}^{(g)} = (\boldsymbol{C}^{(g_1)} + \boldsymbol{C}^{(g_2)})/2$ yields

$$\left(\boldsymbol{m}^{(g_1+1)} - \boldsymbol{m}^{(g_1)}\right)^{\mathrm{T}} \boldsymbol{C}^{(g)^{-1}} \left(\boldsymbol{m}^{(g_2+1)} - \boldsymbol{m}^{(g_2)}\right) \approx 0 \ . \tag{32}$$

Given $1/c_{\mathrm{cov}} \gg 1$ and (29) we assume also $\boldsymbol{p}_\sigma^{(g-1)^{\mathrm{T}}} \boldsymbol{C}^{(g)^{-\frac{1}{2}}}(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}) \approx 0$ and derive

$$\left(\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}\right)^{\mathrm{T}} \boldsymbol{C}^{(g)^{-1}} \left(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}\right) \approx 0 \ . \tag{33}$$

That is, the steps taken by the population mean become approximately $\boldsymbol{C}^{(g)^{-1}}$-conjugate.

Because $\sigma^{(g)} > 0$, (27) is equivalent to

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right) \tag{34}$$

# 3 Invariance

Invariance, in physics referred to as *symmetry*, is a fundamental concept in science. The purpose of invariance is well reflected in the following quote attributed to Albert Einstein:

> The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.

Invariance is the mathematical concept associated with this aim. For example, we desire a physical law or biological model to be invariant to environmental parameters, say weekday, temperature, or air humidity. Inclusion of these parameters into the model or the need for controlling them makes the model more complex and/or less general. The more invariance properties a model exhibits, or the fewer dependencies on exogenous parameters the model reveals, the wider is its applicability and the greater is its predictive power.

The same idea holds for invariance properties of search algorithms. In search, invariance properties induce equivalence classes of objective functions, on which the performance of the search algorithm is identical. Consequently, any result observed on a real world problem, or on a test function, does not only hold for this single problem instance, but inevitably *generalizes to the complete class of problems induced by the invariance property*, thereof the tested problem is an element. Hence stronger statements on the performance of the search algorithm can be made—a greater number of empirical facts is covered.[9]

The drawback to invariance properties in search is that whenever an invariance property is achieved, some information cannot be exploited anymore. For example, rotational invariance means to abandon exploitation of the orientation of the coordinate system and therefore exploitation of separability. We review important invariance properties of search algorithms.

### 3.1 Invariance under Function Value Transformations

First we consider invariance under transformations $\mathcal{T} : \mathbb{R} \to \mathbb{R}$ of the objective function value, specifically for the objective function $f(\boldsymbol{x}) = \mathcal{T}(g(\boldsymbol{x}))$, for all $g : \mathbb{R}^n \to \mathbb{R}$.

- Invariance to adding a constant to the function value, that is $f = \mathcal{T}(g) = g + a$ for $a \in \mathbb{R}$.
- Invariance under scaling of the function value, that is $f = \mathcal{T}(g) = a \times g$ for $a > 0$.

[9] Invariance per se does *not* imply good performance, it *only* provides means *to generalize* favorable (and unfavorable) performance observations.

- Invariance under order preserving transformations of the objective function value, where $\mathcal{T}$ is a strictly monotonically increasing function. Invariance under order preserving transformations includes the above listed invariance properties and is much more general.

Because CMA-ES depends only on *ranking* of function values, it achieves the above listed invariance properties. The sequence of generated search points is independent of $\mathcal{T}$ as given above. We believe that this is a very important feature of comparison-based search methods [26].

### 3.2 Invariance under Search Space Transformations

Let $U : \mathbb{R}^n \to \mathbb{R}^n$ be a transformation of the search space. We consider invariances for the objective function $f(\boldsymbol{x}) = g(U(\boldsymbol{x}))$ under certain transformations $U$, for all $g : \mathbb{R}^n \to \mathbb{R}$. Strictly speaking, invariance under $U$ only holds, if also the *initial conditions* are chosen appropriately. In CMA-ES initial mean and covariance matrix of the search distribution must be chosen accordingly.

**Translation invariance** means invariance under $U(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{a}$ for $\boldsymbol{a} \in \mathbb{R}^n$. Translation invariance must be taken for granted in continuous domain search. Lacking translational invariant must be interpreted as having a *inherent*, problem independent assumption about the location of the optimal solution. For a search algorithm this seems to be a contradiction in terms. For example, if zero is a distinguished solution point for the algorithm, on many test function sets exceptional performance can be achieved, but the result is entirely artificial. In contrast, the initial solution should be interpreted as a *justified*, problem dependent assumption about the location of the optimal solution.

**Scale invariance** means invariance under $U(\boldsymbol{x}) = \alpha\,\boldsymbol{x}$, where $\alpha > 0$. From the algorithm descriptions as given below, and given the appropriate initial conditions, one can easily verified that CMA-ES is scale invariant.

Finally, we have invariance properties where $U(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{A}$ is a full rank matrix.

**Diagonal invariance** is invariance under diagonal linear transformations, *i.e.* under a scaling of variables. The matrix $\boldsymbol{A}$ is diagonal.

**Rotational invariance** is invariance under angle preserving, *i.e.* rigid linear transformations of the search space (rotation, reflection). That is, $\boldsymbol{A}$ is an orthogonal matrix. Rotational invariance is closely related to decomposability and separability (see above), because, in most cases, a separable function becomes non-separable under rotation. Therefore a search algorithm can only *either* exploit separability *or* be rotational invariant.

**General linear invariance** is invariance under any full rank, *i.e.* invertible matrix $\boldsymbol{A}$. This invariance includes rotational and diagonal invariance and requires to abandon any inherent model of isotropy and scales. The CMA-ES is invariant under general linear transformations [37].

We conjecture that the impact of an invariance property is related to the degrees of freedom related to the transformation. Consequently, orthogonal and general linear invariance must be considered important all together with invariance under order preserving transformations of the objective function value.

Most likely in practice, initial parameters cannot be chosen according to a desired search space invariance property. Therefore, an invariant search algorithm must also be adaptive: initial settings, in case with poor performance, must evolve within the iteration sequence into "the invariant" parameter values, preferably with good performance, rendering the algorithm as independent of the initial conditions as possible. Adaptivity has the additional advantage that changes of the optimal parameter values over time, *i.e.* their dependency on the position in search space, can be assimilated[10]. While any time-invariable distribution exhibits general linear invariance, *given the (initial) distribution parameters are chosen appropriately*, the lack of adaptivity renders it rather useless. Adaptivity must be regarded as the essential practical counterpart of any invariance property that depends on the intial conditions.

# 4 Adaptive Encoding

In this section we prove rigorously that the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) can be decomposed into two mainly *independent* components. A comparatively simple evolution strategy with step-size adaptation and an *adaptive linear encoding-decoding* mechanism. The prove shows that the efficient

[10] The most prominent examples are step-sizes which often must decrease by several order of magnitudes during the evolution.

adaptation mechanism of CMA can be applied to any continuous domain search algorithm. This is a very powerful result.

### 4.1 Notations

The following notations are used

$S$ the **state space** of the search algorithm; [$\mathcal{A} : S \to S$] an iteration step of the search algorithm $\mathcal{A}$;

$T_B : S \to S$ an invertible transformation, the **decoding** of the state space, the change of representation. The $T_B$ is parameterized by a matrix $\boldsymbol{B}$ and therefore uniquely depends on $\boldsymbol{B}$;

$\boldsymbol{B} \in \mathbb{R}^{n \times n}$ a full rank matrix, representing (i) a new **coordinate system** and a coordinate system transformation in $\mathbb{R}^n$, and (ii) a problem *representation* and linear *decoding* of candidate solutions $\boldsymbol{B} : \boldsymbol{x} \mapsto \boldsymbol{B}\boldsymbol{x}$;

$\mathcal{U} : \mathbb{R}^{n \times n} \times S \to \mathbb{R}^{n \times n}, (\boldsymbol{B}, s) \mapsto \mathcal{U}(\boldsymbol{B}, s)$ the change of representation by updating the matrix $\boldsymbol{B}$. For convenience, we assume that all necessary information to update $\boldsymbol{B}$ is included in the algorithm state $s$ and we may write $\mathcal{U}(\boldsymbol{B})$ instead of $\mathcal{U}(\boldsymbol{B}, s)$;

$\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \in \mathbb{R}^n$ indicates a $(0, 1)$-normal distribution in each coordinate

$i : \mu$ indicates the index of the $i$-th best solution when evaluated on $f$, for example, on the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\}$ we have $\boldsymbol{x}_{1(f)} = \arg\min_{i=1,\ldots,\mu} \{f(\boldsymbol{x}_i)\}$

### 4.2 Introducing Adaptive Encoding

Adaptive Encoding has been introduced in [31] and is shortly revised in the following. Let $f : \mathbb{R}^n \to \mathbb{R}$ be an objective function to be minimized. A search algorithm proposes new candidate solutions in an iterated procedure and evaluates them on $f$. We denote one iteration step as $\mathcal{A}_f : S \to S$, where $S$ denotes the state space of the algorithm. The iteration step can be surrounded by an encoding-decoding step, where $T_B : S \to S$ is the decoding transformation. Here we assume that $T_B$ is parameterized by an $n \times n$-matrix $\boldsymbol{B}$. The new "encoded" algorithm is defined as

$$\mathcal{A}_f^{T_B} : S \to S \tag{35}$$

$$s \mapsto T_B(\mathcal{A}_{f \circ \boldsymbol{B}}(T_B^{-1}(s))) \tag{36}$$

or

$$\mathcal{A}_f^{T_B} \equiv T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1} \quad . \tag{37}$$

The mapping $T_B$ is a decoding of the algorithms state. If $T_B$ and $\boldsymbol{B}$ are the identity we have $\mathcal{A}_f^{T_B} \equiv \mathcal{A}$. We assume, for convenience and w.l.o.g., that recently evaluated solutions are part of the algorithms state. The matrix $\boldsymbol{B}$ serves as linear decoding map.

**Remark 1 (Evaluation of solutions)** *In order to make use of Eq.* (37)*, we have to ensure that candidate solutions are used in their original representation. The solutions must be decoded for evaluation. In other words,* $\mathcal{A}_{f \circ \boldsymbol{B}}$ *in Eq.* (37) *operates on* $f \circ \boldsymbol{B}$*.*

**Remark 2** *We shall choose* $T_B$ *such that solutions which are part of the state space are decoded by the application of* $\boldsymbol{B}$*. With an abuse of notation, or where the state space consist of a single solution only, we write for solution* $\boldsymbol{x}$ *that* $\boldsymbol{B} T_B^{-1}(\boldsymbol{x}) = \boldsymbol{x}$*.*

Considering Remark 1, we can execute the algorithm $\mathcal{A}$ in any coordinate system of our choice. The new coordinate system, where the operations of $\mathcal{A}_{f \circ \boldsymbol{B}}$ are effectively conducted, is defined by $\boldsymbol{B}$. Optimizing $f \circ \boldsymbol{B}$ instead of $f$ renders $\mathcal{A}$ independent of the *given* coordinate system (if $\boldsymbol{B}$ is chosen independent of the given coordinate system). Eq. (37) becomes particularly meaningful when $\boldsymbol{B}$ is adapted.

Finally, in order to specify the objective of adaptive encoding we assume to have a performance measure when running an algorithm on an objective function $f$. The performance measure determines whether one algorithm is better than another. For example, a typical, quantitatively useful measure is the number of candidate solutions evaluated on $f$ until a target function value is achieved.

Equation (37) represents an iteration step of a search algorithm with an additional encoding-decoding procedure. The encoding is parameterized by a $n \times n$-matrix; it therefore adds $n^2$ degrees of freedom. Obviously, the idea is to find a good encoding for algorithm $\mathcal{A}$.

**Aim 1 (static encoding)** *The goal of finding a good encoding is to find a transformation* $T_B$*, such that*

$$T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1} \textit{ outperforms } \mathcal{A}_f$$

Because the transformation $T_B$ is static, it needs to be applied only once before the first iteration, and when solutions are evaluated or delivered. Therefore, the static encoding is usually part of the design of the objective function while $\mathcal{A}_f$ is applied.

The static formalism of Aim 1 is indeed not very interesting. To get a more interesting situation, we need to consider an *update* or *adaptation* of the encoding $T_B$.

**Definition 1** *(Adaptive Encoding) Given an algorithm, $\mathcal{A}$, an encoding, $T_B$, parameterized with $\boldsymbol{B}$, and an update, $\mathcal{U}$, the iteration step of an adaptively encoded algorithm in state $s \in S$ is defined as*

$$s \leftarrow T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1}(s) \tag{38}$$

$$\boldsymbol{B} \leftarrow \mathcal{U}(\boldsymbol{B}, s) \tag{39}$$

*where $\leftarrow$ denotes the assignment operator and $T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1}(s) = T_B(\mathcal{A}_{f \circ \boldsymbol{B}}(T_B^{-1}(s)))$. We write $T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1}\,;\, \mathcal{U}(T_B)$ to denote the iteration step of Equations (38) and (39).*

Obviously, any iterative algorithm $\mathcal{A}$ can be plugged into the adaptive encoding mechanism.

**Proposition 1** *(Adaptive Encoding is universal) The Adaptive Encoding from Definition 1 can be applied to any search algorithm—any search algorithm can be adaptively encoded.*

**Proof** The proposition follows directly from the definition of $T_B$ as invertible mapping from $S$ to $S$. □

Even though Proposition 1 is just about trivial, it is of utmost importance for the implications of our results, because it establishes the general applicability of any effective adaptive encoding.

Analogous to Aim 1, we consider the merits of an adaptive encoding.

**Aim 2 (adaptive encoding)** *Find a transformation $T$ and an update $\mathcal{U}$, such that for a given $\boldsymbol{B}'$ and a given initial $\boldsymbol{B}$*

$$T_B \circ \mathcal{A}_{f \circ \boldsymbol{B}} \circ T_B^{-1}\,;\, \mathcal{U}(T_B) \text{ outperforms } T_{B'} \circ \mathcal{A}_{f \circ \boldsymbol{B}'} \circ T_{B'}^{-1}.$$

*The left iteration step updates the encoding, the right iteration step applies a constant encoding, $T_{B'}$, to algorithm $\mathcal{A}$.*

Taking only a single iteration step, Aim 2 does not depend on the update $\mathcal{U}$ and it reduces to Aim 1. Consequently, Aim 2 becomes only interesting, when an iteration *sequence* is considered. Indeed, in a realistic automated scenario, an adaptation can only be achieved in the iteration sequence.

Finally, we define three cases/scenarios when considering Aim 2.

**Scenario 1** *(Standard scenario) The initial $\boldsymbol{B}$ equals $\boldsymbol{B}'$. Aim 2 shall be satisfied for most given $\boldsymbol{B}'$.*

The standard scenario applies in particular if there exists an optimal $\boldsymbol{B}_{\mathrm{opt}}$, that does not depend on the position in search space. In this case, for $\boldsymbol{B}' = \boldsymbol{B}_{\mathrm{opt}}$, we cannot outperform $T_{B'} \circ \mathcal{A}_{f \circ \boldsymbol{B}'} \circ T_{B'}^{-1}$.

**Scenario 2** *(Ambitious scenario) The initial $\boldsymbol{B}$ equals $\boldsymbol{B}'$. Aim 2 shall be satisfied* for all *given $\boldsymbol{B}'$.*

Satisfying the ambitious scenario implies that no fixed optimal encoding exists and a changing encoding can be advantageous compared to any fixed encoding. Both, the standard and the ambitious scenario are reasonable objectives, depending on the given objective function.

**Scenario 3** *(Unrealistic scenario) For all initial $\boldsymbol{B} \neq \boldsymbol{B}'$, Aim 2 shall be satisfied.*

As we might be able to choose $\boldsymbol{B}$ arbitrarily bad, it seems unrealistic that Aim 2 can be satisfied for any initial $\boldsymbol{B} \neq \boldsymbol{B}'$ in general.

In the following we propose and investigate an effective way to implement adaptive encoding as given in Definition 1.

### 4.3 An Adaptive Encoding Procedure: $\mathrm{AE}_{\mathrm{CMA}}$

For a thorough definition of an adaptively encoded search procedure we need to define $T_B$ and $\mathcal{U}$. The choice of the transformation $T_B$ depends on the state space of the algorithm to which adaptive encoding is applied and is usually not very involved. Here, at first, we present a generic update $\mathcal{U}$. The procedure is given in Procedure 1 $\mathrm{AE}_{\mathrm{CMA}}$-Update[11] (p. 29). The procedure takes as input a number of solution points, sorted according to their objective function value, and a matrix $\boldsymbol{B}$. The matrix $\boldsymbol{B}$ is updated and considered as "output" of the procedure. Three main equations of the update procedure are derived from the CMA-ES, as referenced in the procedure.

[11] Matlab code of $\mathrm{AE}_{\mathrm{CMA}}$-Update is provided in `http://www.lri.fr/~hansen/AEupdate.m`

**Procedure 1:** $\mathtt{AE_{CMA}\text{-}Update}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{B})$

updates the encoding matrix $\boldsymbol{B}$ using the $\mu$ recent best-ranked candidate solutions

given parameters $w_i, c_\mathrm{p}, c_1, c_\mu$
given $\boldsymbol{m} \in \mathbb{R}^n$ and $\boldsymbol{p} \in \mathbb{R}^n$ from last iteration, or $\boldsymbol{p} = \boldsymbol{0}$, $\boldsymbol{m} = \boldsymbol{x}_1$
let matrices $\boldsymbol{B}^\circ$ orthogonal, and $\boldsymbol{D}$ diagonal, with diagonal elements sorted in ascending order , "$\leftarrow$" assigns accordingly
$\boldsymbol{m}^- = \boldsymbol{m}$
$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_i$ `// Eq.` (4)
set scalars $\alpha_i \geq 0$, for $i = 0, \ldots, \mu$, see text
$\boldsymbol{p} \leftarrow (1 - c_\mathrm{p})\, \boldsymbol{p} + \sqrt{c_\mathrm{p}\,(2 - c_\mathrm{p})}\, \alpha_0 (\boldsymbol{m} - \boldsymbol{m}^-)$ `// Eq.` (20)
$\boldsymbol{C}_\mu = \sum_{i=1}^{\mu} w_i\, \alpha_i^2\, (\boldsymbol{x}_{i:\mu} - \boldsymbol{m}^-)(\boldsymbol{x}_{i:\mu} - \boldsymbol{m}^-)^\mathrm{T}$
set scalar $\alpha_\mathrm{p} \geq 0$, see text
$\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu)\, \boldsymbol{B}\boldsymbol{B}^\mathrm{T} + c_1 \alpha_\mathrm{p}\, \boldsymbol{p}\boldsymbol{p}^\mathrm{T} + c_\mu \boldsymbol{C}_\mu$ `// Eq.` (25)
$\boldsymbol{B}^\circ \boldsymbol{D} \boldsymbol{D} \boldsymbol{B}^\circ \leftarrow \boldsymbol{C}$ `// eigendecomposition`
optionally normalize $\boldsymbol{D}$
$\boldsymbol{B} \leftarrow \boldsymbol{B}^\circ \boldsymbol{D}$ `// encoding matrix`

**Proposition 2** *Let $\sigma$ denote a step-size and $\mu_\mathrm{eff}^{-1} = \sum_{i=1}^{\mu} w_i^2$. Let $\alpha_0 = \frac{\sqrt{\mu_\mathrm{eff}}}{\sigma}$, $\alpha_i = \sigma^{-1}$, for $i = 1, \ldots, \mu$, $\alpha_p = 1$. Then, AE$_{CMA}$-Update implements the update equations for the evolution path, $\boldsymbol{p}_\mathrm{c}^{(g)}$, and the covariance matrix, $\boldsymbol{C}^{(g)} = \boldsymbol{B}\boldsymbol{B}^\mathrm{T}$, in the $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES.*

**Proof** Assuming that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu$ are the $\mu$ best solutions in the recent iteration step, line 5 computes $\boldsymbol{m}$ according to Eq. (4). Lines 7 and 10 replicate the covariance matrix update equations (20) and (25) with added or renamed normalization coefficients, denoted as $c$ and $\alpha$. Substituting the coefficients results in the original equations. □

A slow change of $\boldsymbol{B}$ might be desirable. While $\boldsymbol{C}$ will only change slowly, as long as $c_1$ and $c_\mu$ are small, the decomposition of $\boldsymbol{C}$ does not ensure a similar behavior for $\boldsymbol{B}^\circ$ and $\boldsymbol{D}$. For this reason, the diagonal elements are sorted in $\boldsymbol{D}$. We conjecture, that lines 11 to 13 can be replace by a cholesky decomposition, or, more promising, by an incremental cholesky update, similar to [77], that ensures small changes, as long as $c_1$ and $c_\mu$ are small. In this case, it might be sufficient to only encode the solutions for the function evaluation and, as an approximation, completely abandon the encoding-decoding of the algorithms state.

#### 4.3.1 Choice of Parameters

In Procedure 1 AE$_{\text{CMA}}$-Update, the scalars $\alpha_{\text{p}}$ and $\alpha_i$ for $i = 0, \ldots, \mu$, need to be chosen. They normalize the input entries for the covariance matrix update (most of them are the difference between a new solution and the former mean). In the original CMA, we can derive the expected lengths of the input entries from the sampling procedure. Under random selection the normalized entries are distributed according to

$$\mathcal{N}(\mathbf{0}, \boldsymbol{C}) \quad = \quad \boldsymbol{B}\,\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \quad . \tag{40}$$

In general, we cannot assume to know the expected lengths of the input entries, therefore we need to normalize them. In Eq. (40), the expected squared length of the decoded input entry, $E\|\boldsymbol{B}^{-1}\boldsymbol{B}\,\mathcal{N}(\mathbf{0}, \boldsymbol{I})\,\|^2$, computes to $n$ suggesting a normalization to length $\sqrt{n}$. Keeping this in mind, we discuss the choice of the scalar coefficients in turn.[12]

$\alpha_0 = \frac{\sqrt{n}}{\|\boldsymbol{B}^{-1}(\boldsymbol{m} - \boldsymbol{m}^-)\|}$ , normalizes the difference $\boldsymbol{B}^{-1}(\boldsymbol{m} - \boldsymbol{m}^-)$ to length $\sqrt{n}$. Consequently, only the direction is relevant and the absolute size of the difference is disregarded.

$\alpha_i = \frac{\sqrt{n}}{\|\boldsymbol{B}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}^-)\|}$ , for $i = 1, \ldots, \mu$, is the conservative choice, where the length of the difference $\boldsymbol{B}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}^-)$ is disregarded. In general, we recommend to choose

$$\alpha_i = \frac{\sqrt{n}}{\max\left(\frac{l_i}{\beta}, \underset{j=1,\ldots,\mu}{\text{median}}\,(l_j)\right)} \qquad \text{for } i = 1, \ldots, \mu \ , \tag{41}$$

where $l_i = \|\boldsymbol{B}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}^-)\|$. In this way, the median is set to "length" $\sqrt{n}$ and the maximal length is set to $\beta\sqrt{n}$ with $\beta \geq 1$. We recommend $\beta = 2$. Unusual large entries may, for example, occur if solutions are originally sampled from a distribution with heavy tails. By chance, an outranging solution could enter the procedure despite a bad objective function value and an unjustified very large change of $\boldsymbol{B}$ would result.

$\alpha_{\mathbf{p}} = 1$ will be the usual choice, while $\alpha_{\text{p}} = \frac{\sqrt{n}}{\|\boldsymbol{p}\|}$ is a conservative alternative and will not allow to utilize the evolution path effectively; $\alpha_{\text{p}} = 0$ would be even more conservative.

[12] We ignore the case of denominators being zero, where the respective coefficient $\alpha$ can be set to any positive number.

Finally, we give the default settings for the constants used in CMA-ES-Update and discuss the choices in turn.

$c_{\mathbf{p}} = \frac{1}{\sqrt{n}}$ is the learning constant for the evolution path, which should be usually between $\frac{1}{\sqrt{n}}$ and $\frac{2}{n+1}$ [37]. For larger $c_{\mathrm{p}}$, the effect of the evolution path will attenuate. The backward time horizon for the evolution path is roughly ${c_{\mathrm{p}}}^{-1}$. We choose as default the "conservative" limit of the useful range, *i.e.* a comparatively large $c_{\mathrm{p}}$.

$w_i = \frac{\ln(\mu+1)-\ln i}{\mu\ln(\mu+1)-\sum_{j=1}^{\mu}\ln j}$ , for $i = 1, \ldots, \mu$ are the recombination weights. They (must) sum to one and obey $w_1 \geq \cdots \geq w_\mu \geq 0$. Generally, we choose $\mu$ being half of the overall generated number of solutions per iteration (before selection).

$c_1 = \frac{\alpha_{\mathbf{cov}}}{(n+1.3)^2+\mu_{\mathrm{eff}}}$ is the learning rate for the rank-one update in line 10 of CMA-ES-Update (middle summand), with $\alpha_{\mathrm{cov}} = 0.2$ as default. The denominator being quadratic in $n$ reflects the degrees of freedom in the encoding matrix $\boldsymbol{B}$. The formula is derived as a simplification from the original formulation in [30].

$c_\mu = \alpha_{\mathbf{cov}} \frac{\mu_{\mathrm{eff}} - 2 + \frac{1}{\mu_{\mathrm{eff}}}}{(n+2)^2+\alpha_\mu \mu_{\mathrm{eff}}}$ is the learning rate for the rank-$\mu$ update (right summand in line 10), with $\alpha_{\mathrm{cov}} = 0.2$ and $\alpha_\mu = 0.2$ as default. With increasing $\mu_{\mathrm{eff}}$, the learning rate increases and gets close to one.

$\alpha_{\mathbf{cov}} = 0.2$ must be chosen positive and such that $c_1 + c_\mu \leq 1$. The default value of $0.2$ is about ten times smaller, *i.e.* considerably more conservative, than for CMA-ES. Too large values for $\alpha_{\mathrm{cov}}$ potentially lead to a failure. Too small values slow down the adaptation. At least a minimalistic parameter study for $\alpha_{\mathrm{cov}}$ is recommended.

The final parameter setting needs to be decided specifically for a given algorithm. We believe that the given guidelines will be usually sufficient to find good settings with reasonable effort. To our experience, a good setting works across many objective functions and the identification needs to be conducted only once on a few simple test functions.

#### 4.3.2 Application of AE$_{\mathbf{CMA}}$-Update to CSA-ES

We apply the introduced update procedure to an evolution strategy with *cumulative step-size adaptation* [58, CSA-ES].

**CSA-ES** Algorithm 2 implements the CSA-ES. The main iteration step of the algorithm is outlined in Procedure 3.

**Algorithm 2**: $(\mu/\mu_{\mathrm{W}}, \lambda)$-CSA-ES

given fitness $f$
initialize population $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \in \mathbb{R}^n$
initialize step-size $\sigma > 0$
initialize evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$
**repeat**
    CSA-ES-Step$(f, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \sigma, \boldsymbol{p}_\sigma)$
**until** *stopping criterion is met*

**Procedure 3**: `CSA-ES-Step(`$f, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \sigma, \boldsymbol{p}_\sigma$`)`

given parameters $w_i, c_{\mathrm{p}}$
$\boldsymbol{m}^- = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_i$
$\boldsymbol{x}_i \leftarrow \boldsymbol{m}^- + \sigma \, \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \,, \quad$ for $i = 1, \ldots, \lambda$
evaluate $\boldsymbol{x}_i$ on $f \rightarrow \boldsymbol{x}_{i:\mu} \,, \quad$ for $i = 1, \ldots, \lambda$
$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i:\mu} \,, \quad$ for $i = 1, \ldots, \lambda$ `// sorted`
$\boldsymbol{m} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_i$
$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma) \, \boldsymbol{p}_\sigma + \sqrt{c_\sigma \, (2 - c_\sigma) \mu_{\mathrm{eff}}} \, \frac{1}{\sigma} (\boldsymbol{m} - \boldsymbol{m}^-)$
$\sigma \leftarrow \sigma \, \exp\left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\mathbf{0}, \boldsymbol{I})\|} - 1 \right) \right)$

**$\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES** For applying adaptive encoding to CSA-ES, we choose the following invertible encoding for the state variables in CSA-ES,

$$T_B : ((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{p}_\sigma, \sigma) \mapsto ((\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu), \boldsymbol{B}^\circ \boldsymbol{p}_\sigma, \sigma) \ . \tag{42}$$

Besides $\boldsymbol{B}$ for encoding the solutions $\boldsymbol{x}_i$, the orthogonal matrix $\boldsymbol{B}^\circ$ (*i.e.* $\boldsymbol{B}$ with normalized columns) is used for encoding the evolution path $\boldsymbol{p}_\sigma$, see also Section 2.4.

Finally, in Algorithm 4, adaptive encoding is applied to CSA-ES, where $\mathrm{AE}_{\mathrm{CMA}}$-Update is used to update the encoding.

**Algorithm 4**: $\text{AE}_{\text{CMA}}$-CSA-ES
Shaded areas add to CSA-ES and implement the adaptive encoding, $\text{AE}_{\text{CMA}}$

given fitness $f$
initialize population $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \in \mathbb{R}^n$
initialize step-size $\sigma > 0$
initialize evolution path $\boldsymbol{p}_\sigma = \boldsymbol{0}$
initialize encoding matrix $\boldsymbol{B} = \boldsymbol{I}$
**repeat**
  $T_B^{-1}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{p}_\sigma)$
  CSA-ES-Step($f \circ \boldsymbol{B}$, $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \sigma, \boldsymbol{p}_\sigma$)
  $T_B((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{p}_\sigma)$
  $\text{AE}_{\text{CMA}}$-Update$((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{B})$
**until** *stopping criterion is met*

### 4.4 $\text{AE}_{\text{CMA}}$-CSA-ES Recovers CMA-ES

We show now that the adaptive encoding $\text{AE}_{\text{CMA}}$ recovers the original CMA-ES when it is applied to CSA-ES as in Algorithm 4. For practical purpose we rewrite the CMA-ES from Section 2 in Algorithm 5 and Procedure 6.

**Algorithm 5**: CMA-ES

given fitness $f$
initialize population $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \in \mathbb{R}^n$
initialize $\sigma > 0$ (step-size)
initialize evolution paths $\boldsymbol{p} = \boldsymbol{p}_\sigma = \boldsymbol{0}$
initialize $\boldsymbol{C} = \boldsymbol{I}$
**repeat**
  CMA-ES-Step($f, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{C}, \sigma, \boldsymbol{p}, \boldsymbol{p}_\sigma$)
**until** *stopping criterion is met*

We now proof that Algorithm 4 and Algorithm 5 are identical, that is, if $\text{AE}_{\text{CMA}}$ is applied to CSA-ES the CMA-ES is recovered.

**Theorem 1 (Recovery of CMA-ES)** *Given $T_B$ as in Eq. (42) and the scalars for $\text{AE}_{\text{CMA}}$-Update in each iteration as given in Proposition 2, then the CMA-ES-*

**Procedure 6:** `CMA-ES-Step(`$f, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \boldsymbol{C}, \sigma, \boldsymbol{p}, \boldsymbol{p}_\sigma$`)`

1 given parameters $w_i, c_\sigma, c_\mathrm{p}, c_1, c_\mu$
2 let matrices $\boldsymbol{B}^\circ$ orthogonal, and $\boldsymbol{D}$ diagonal, with diagonal elements sorted in ascending order
3 $\boldsymbol{m}^- = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_i$ `// previous population mean`
4 $\boldsymbol{B}^\circ \boldsymbol{D} \boldsymbol{D} \boldsymbol{B}^{\circ\mathrm{T}} = \boldsymbol{C}$ `// eigendecomposition`
5 $\boldsymbol{C}^{-1/2} = \boldsymbol{B}^\circ \boldsymbol{D}^{-1} \boldsymbol{B}^{\circ\mathrm{T}}$
6 $\boldsymbol{x}_i \leftarrow \boldsymbol{m}^- + \sigma \boldsymbol{B}^\circ \boldsymbol{D} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\,, \quad$ for $i = 1, \ldots, \lambda$
7 evaluate $\boldsymbol{x}_i$ on $f$, $\quad$ for $i = 1, \ldots, \lambda$
8 $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i:\mu}\,, \quad$ for $i = 1, \ldots, \lambda$ `// sorted`
9 $\boldsymbol{m} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_i$
10 $\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma) \, \boldsymbol{p}_\sigma + \sqrt{c_\sigma \, (2 - c_\sigma) \mu_\mathrm{eff}} \, \boldsymbol{C}^{-1/2} \frac{1}{\sigma} (\boldsymbol{m} - \boldsymbol{m}^-)$
11 $\boldsymbol{p} \leftarrow (1 - c_\mathrm{p}) \, \boldsymbol{p} + \sqrt{c_\mathrm{p} \, (2 - c_\mathrm{p}) \mu_\mathrm{eff}} \, h(\boldsymbol{p}_\sigma) \frac{1}{\sigma} (\boldsymbol{m} - \boldsymbol{m}^-)$
12 $\boldsymbol{C}_\mu = \sum_{i=1}^{\mu} w_i \, \frac{1}{\sigma^2} (\boldsymbol{x}_i - \boldsymbol{m}^-)(\boldsymbol{x}_i - \boldsymbol{m}^-)^\mathrm{T}$
13 $\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu) \, \boldsymbol{C} + c_1 \boldsymbol{p}\boldsymbol{p}^\mathrm{T} + c_\mu \boldsymbol{C}_\mu$
14 $\sigma \leftarrow \sigma \, \exp\left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\|} - 1 \right) \right)$

*$(\mu/\mu_\mathrm{w}, \lambda)$-CSA-ES (Algorithm 4) implements the $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES (Algorithm 5).*

**Proof** We assume the same (initial) state for $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu), \sigma, \boldsymbol{C}, \boldsymbol{p}_\sigma, \boldsymbol{p}$ in $\mathrm{AE}_\mathrm{CMA}$-$(\mu/\mu_\mathrm{w}, \lambda)$-CSA-ES and $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES after line 6 in both algorithms and refer to this state as *loop entry*. We investigate one iteration step.

First we consider the sampled evaluated solutions $\boldsymbol{x}_i$, for $i = 1, \ldots, \lambda$, in CSA-ES-Step. Because the solutions are evaluated on $f \circ \boldsymbol{B}$ we consider $\boldsymbol{B}\boldsymbol{x}_i$. We have

$$\begin{aligned} \boldsymbol{B}\boldsymbol{x}_i &\leftarrow \boldsymbol{B}(\boldsymbol{m}^- + \sigma \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})) \\ &= \boldsymbol{B} \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_i + \sigma \boldsymbol{B} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \\ &= \sum_{i=1}^{\mu} w_i \, \boldsymbol{B}\boldsymbol{x}_i + \sigma \boldsymbol{B}^\circ \boldsymbol{D} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \end{aligned} \tag{43}$$

With the definition of $T_B$ we find that $\boldsymbol{x}_i$ in CSA-ES-Step equals $\boldsymbol{B}^{-1}\boldsymbol{x}_i$ at the loop entry. Therefore the new solutions evaluated on $f$ according to (43) are the same as those in line 6 of CMA-ES-Step. The new solutions $\boldsymbol{B}\boldsymbol{x}_i$ in CSA-ES-Step correspond to the new solutions $\boldsymbol{x}_i$ in CMA-ES-Step. Because $T_B(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{x}$ we

find that $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu)$ at the loop end is identical in both cases. Consequently, with the linearity of $\boldsymbol{B}$, we recover $\boldsymbol{m}$ and $\boldsymbol{m}^-$ from lines 3 and 9 in Procedure 6 CMA-ES-Step by $\boldsymbol{B}\boldsymbol{m}$ and $\boldsymbol{B}\boldsymbol{m}^-$ from lines 2 and 6 in Procedure 3 CSA-ES-Step.

It remains to be shown that AE$_{\text{CMA}}$-CSA-ES recovers the update of $\boldsymbol{p}_\sigma$, $\sigma$, $\boldsymbol{p}$ and $\boldsymbol{C}$ in CMA-ES. We treat each variable in turn.

We investigate the evolution path $\boldsymbol{p}_\sigma$ for the step-size in CSA-ES-Step which is transformed with $\boldsymbol{B}^\circ$ before the loop end.

$$
\begin{aligned}
T_B(\boldsymbol{p}_\sigma) &= \boldsymbol{B}^\circ \boldsymbol{p}_\sigma && (44)\\
&\leftarrow \boldsymbol{B}^\circ \left( (1-c_\sigma)\, \boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2-c_\sigma)\mu_{\text{eff}}}\, \frac{1}{\sigma}(\boldsymbol{m} - \boldsymbol{m}^-) \right) \\
&= (1-c_\sigma)\, \boldsymbol{B}^\circ \boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2-c_\sigma)\mu_{\text{eff}}}\, \frac{1}{\sigma} \boldsymbol{B}^\circ (\boldsymbol{m} - \boldsymbol{m}^-) && (45)
\end{aligned}
$$

The $\boldsymbol{B}^\circ \boldsymbol{p}_\sigma$ equals $\boldsymbol{p}_\sigma$ at the loop entry. Using $\boldsymbol{m}$ and $\boldsymbol{m}^-$ from the original coordinate system, we compute the rightmost term of (45) to

$$
\begin{aligned}
\boldsymbol{B}^\circ (\boldsymbol{B}^{-1}\boldsymbol{m} - \boldsymbol{B}^{-1}\boldsymbol{m}^-) &= \boldsymbol{B}^\circ \boldsymbol{B}^{-1} (\boldsymbol{m} - \boldsymbol{m}^-) \\
&= \boldsymbol{B}^\circ (\boldsymbol{B}^\circ \boldsymbol{D})^{-1} (\boldsymbol{m} - \boldsymbol{m}^-) \\
&= \boldsymbol{B}^\circ \boldsymbol{D}^{-1} \boldsymbol{B}^{\circ \mathrm{T}} (\boldsymbol{m} - \boldsymbol{m}^-)\ , && (46)
\end{aligned}
$$

where $\boldsymbol{B} = \boldsymbol{B}^\circ \boldsymbol{D}$. Therefore, Equation (45) recovers the update rule for the evolution path $\boldsymbol{p}_\sigma$ in CMA-ES (line 10 in CMA-ES-Step).

Because $\boldsymbol{B}^\circ$ is orthogonal we have $\|\boldsymbol{B}^\circ \boldsymbol{p}_\sigma\| = \|\boldsymbol{p}_\sigma\|$ and consequently, the step-size update is identical in both cases.

Finally, completing the proof, $\boldsymbol{p}$ and $\boldsymbol{C}$ are updated in line 10 of AE$_{\text{CMA}}$-CSA-ES according to the CMA-ES as shown in Proposition 2. □

Theorem 1 supports the hypothesis that AE$_{\text{CMA}}$-Update is an efficient way to update the representation matrix $\boldsymbol{B}$, as CMA-ES is known to efficiently adapt the principle axes of the coordinate system, where the independent sampling takes place most efficiently. First empirical evidence has been given in [31].

## 5 Perspectives

In this section a few directions of future research perspectives for stochastic variable metrics methods are presented.

## 5.1 Variable Metrics Methods

In the present thesis PCA-based learning and adaptation in stochastic search algorithms has been discussed. Here, we outline two lines of possible future research directions for variable metrics methods.

### 5.1.1 Large scale optimization

The methods discussed in this thesis estimate or adapt $\propto n^2$ internal parameters. The application of such methods becomes of limited benefit in *large-scale* optimization with typically more than a hundred or even thousands of continuous parameters. General limitations stem from the fact that (a) the general adaptation time of these methods amount to $\propto n^2$ function evaluations and (b) the internal computational costs are at least $\propto n^2$ for each function evaluation. Both facts limit the application of such methods to large scale optimization.

In order to address large problem sizes, the degrees of freedom in the sampled search distribution should be reduced. A first, very preliminary step in this direction was taken in [66], where no parameter dependencies are modeled. However, essential dependencies between parameters need to be captured in order to solve non-trivial problems. For example, in the spirit of *dimensionality reduction*, only a few large components might be learned. Respective work was previously proposed for a *single* large component [38][59], or for several components [50], while the latter only aimed to reduce the internal time and space complexity of the algorithm. Overall, there is a clear lack of methods, that perform comparably well as those presented in this thesis, but show this performance only on a subset of less complex functions. Compared to a PCA, these mechanisms will be less general, but achieve a better scalability with the search space dimension.

Building blocks for steps in this research direction are *principal component learning* [57] and *minor component learning* [54][55], well-know machine learning techniques that must be customized for their incorporation into stochastic search algorithms. Based on PCA and minor component analysis, also extreme component analysis [79] could be explored. We expect that it should be possible to reduce the learning time for *still complex landscapes* by approximately one order of magnitude from $\propto n^2$ to $\propto n$.

### 5.1.2 Non-linear models

The second line of future research aims at learning *non-linear models*. This is, in a sense, the opposite direction compared to the previous research line. Non-linear

models generally contain more parameters than linear ones. Therefore, the degrees of freedom for non-linear models must be carefully restricted.

First attempts along this line were taken in [84], [14], [61] using *independent component analysis* (ICA). In evolutionary computation, PCA can efficiently render a *mutation* operator independent of the coordinate system [31]. Similarly, ICA can be used to render a *recombination* or cross-over operator independent of the coordinate system. This can, for example, be useful to exploit the Cauchy distribution, or other non-isotropic heavy-tail distributions, on *non-separable* functions. The idea is appealing, but massive parallelization, for example in a computer grid, might be necessary to apply ICA successfully in a realistic time-scale on non-toy problems. Nevertheless, this approach has the potential to push the feasibility of multi-modal optimization beyond its current limits.

A second approach to non-linear extensions of variable metrics methods builds on kernel-PCA [71, 60] and the construction of *principle curves* [39] using the polygonal line algorithm [48] for stochastic optimization. In this perspective two important steps can be identified: i) the use or development of *incremental* learning methods; ii) the identification of a realistic testbed requiring the use of non-linear kernel-PCA.

### 5.2 Evaluation of Black-Box Optimization Algorithms

Evaluation of optimization algorithms on benchmark function sets is inevitable, before their application to real world problems. Several problems arise in this context. First, a benchmark function set needs to be defined. Many function sets are available [19], [40], [56], [74], [17], [74], [76]. While many of them suffer from obvious deficiencies, useful guidelines for constructing test functions were given, for example, in [81]. Our remaining main concern is whether benchmark functions reflect "the reality" and many common functions are indeed too easy to solve. Second, the experimental and evaluation procedure and, in particular, a well-founded collection, elaboration, presentation, and interpretation of the generated data is a non-trivial, and tedious task. The common practice is often insufficient and was criticized, for example, in [22], [9].

In this line of research we have recently initiated a first achievement (joined work with Anne Auger, Raymond Ros, Marc Schoenauer) in collaboration with the Vorarlberg University of Applied Science (Prof. Hans-Georg Beyer and Steffen Finck), we develop the platform *COmparing Continuous Optimizers*, COCO[13].

[13] see [http://coco.gforge.inria.fr/doku.php]

The main objective is to largely automatize the collection, reporting, presentation, and analysis of data resulting from running stochastic and non-stochastic optimizers on test or real world problems. Compared to other platforms (COCONUT, COIN-OR, CUTEr, LIBOPT, OAT, OpenDP, PISA), our focus is the automated data post-processing and *data analysis and comparison* for single objective optimization. Experimental design and descriptive statistics are essential parts thereof. We foresee a great impact from this work which comprises

- Design of a testbed and the **characterization of problems** and their difficulties. Test functions must be *simple enough*, in order to draw conclusions from the test results, but also *challenging* for the algorithms. We want to achieve both objectives providing a testbed with well characterized *problem classes* and difficulties.
- A unified, carefully conducted **experimental design** greatly strengthens experimental results and also eases performing new experiments. We aim at incrementally collecting comparable performance data for many optimization procedures over years. As data will be produced under the same experimental design and collected in a comparable way—not necessarily with COCO—they will support rigorous and significant performance comparisons.
- Finding good and generally applicable **performance indicators** in optimization is a non-trivial task. Comparing the mean final objective function values and success probabilities is the standard approach, which is unsatisfactory and by no means sufficient. Useful performance indicators can also be graphical [20]. We will identify important criteria for good indicators, apply the most meaningful indicators and develop new indicators [7] according to the given criteria.
- Graphical data presentation will be designed in a principled and ergonomic way. An appropriate presentation of data is an essential part of a meaningful interpretation of results. The platform will include extensive facilities for statistical analysis and hypothesis testing as well as boot-strapping [21] to attain dispersion measures for aggregated performance indicators [7][24].

The organization of a workshop, BBOB 2009[14] at the ACM-GECCO conference 2009 is a first milestone in this long-term project. Figure 6 shows an example of data presentations in the workshop.

[14] see [http://coco.gforge.inria.fr/doku.php?id=bbob-2009]

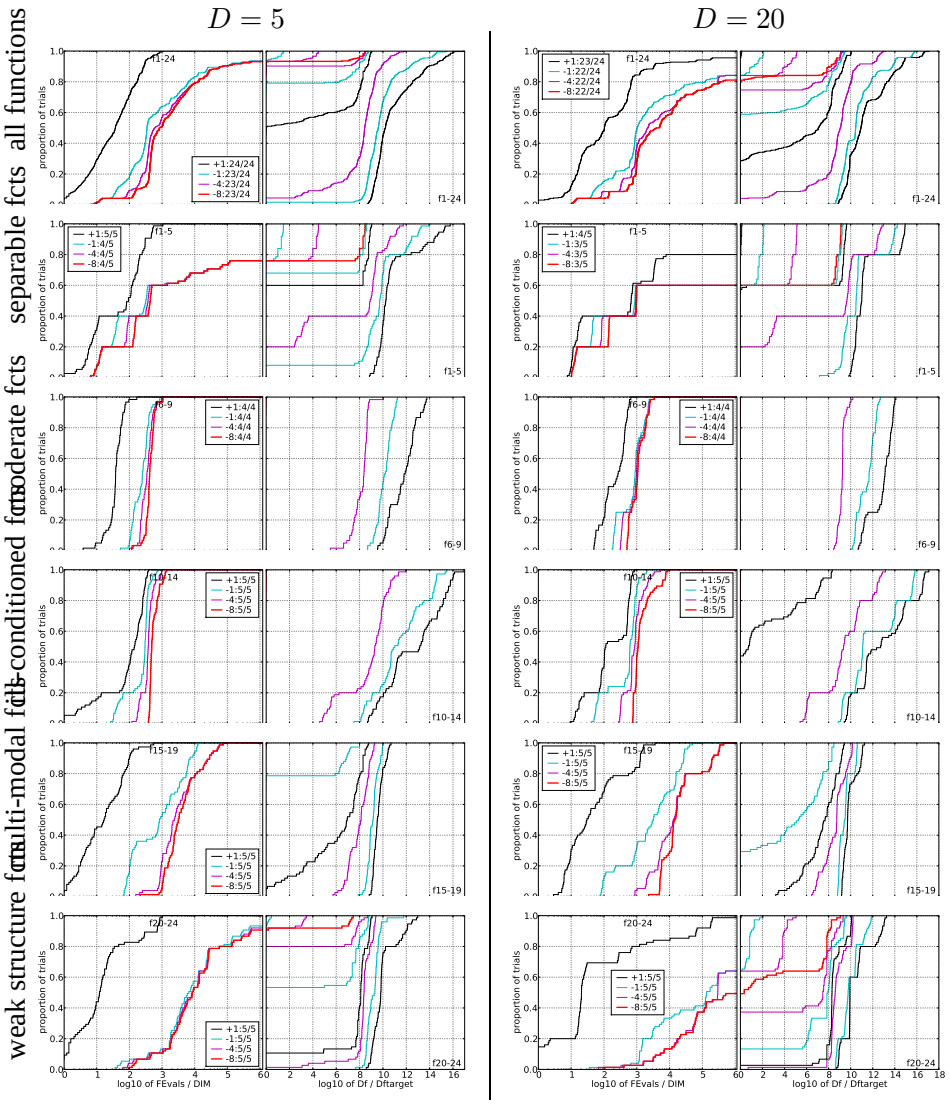


Figure 6: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials versus running time (left subplots) or versus $\Delta f$ (right subplots). The thick red line represents the best achieved results. Left subplots: ECDF of the running time (number of function evaluations), divided by search space dimension $D$, to fall below $f_{\mathrm{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k$ is the first value in the legend. Right subplots: ECDF of the best achieved $\Delta f$ divided by $10^k$ (upper left lines in continuation of the left subplot), and best achieved $\Delta f$ divided by $10^{-8}$ for running times of $D, 10\,D, 100\,D \ldots$ function evaluations (from right to left cycling black-cyan-magenta). Top row: all results from all functions; second row: separable functions; third row: misc. moderate functions; fourth row: ill-conditioned functions; fifth row: multi-modal functions with adequate structure; last row: multi-modal functions with weak structure. The legends indicate the number of functions that were solved in at least one trial. FEvals denotes number of function evaluations, $D$ and DIM denote search space dimension, and $\Delta f$ and Df denote the difference to the optimal function value.

## 5.3 Optimization under Uncertainties and Dynamic Environments

Real world data, as for example measurements from physical experiments or (stochastic) simulations of physical models, are typically subject to noise and uncertainties. Not only is the outcome of the simulation a distribution rather than a single value, also this distribution can change in time resulting in a *dynamic* environment. A typical example is a calibration task, where the true optimum for the parameters might (slowly) change over time. In uncertain and dynamic environments special care must be taken to prevent undesired convergence or divergence of a search algorithm. Population-based stochastic search algorithms are robust against uncertainties [2, 47]. They do not rely, for example, on measurements of small finite differences which are very sensitive to rugged search landscapes. However, typical noise handling techniques are usually expensive, in that they increase the number of function evaluations, or increase the population size, and are therefore less attractive in dynamic environments.

Based on a noise handling method introduced in [35], that can prevent any rank-based search algorithm from converging prematurely, different approaches to improve the efficiency of noise-resistant search algorithms are conceivable. Integrating surrogate model approaches [46] into noise resistant algorithms is a promising approach. In particular, when surrogate methods are based on regression, they can be insensitive to noise. Usually, appropriate assumptions on the noise distribution must be taken. As a second step, rank-based surrogate methods [69] should be less sensitive to the underlying noise distribution. They will also preserve invariance under order-preserving transformation of the objective function value of an underlying rank-based search algorithm.

## 5.4 Theory of Stochastic Search

The analysis of stochastic search algorithms in continuous domain is still in its infancies. In particular, quantitative convergence results and bounds are rare [44][78] or rely on a deterministic model of the underlying algorithm [75]. We believe, an important future direction will emphasize on stochastic stability of algorithms and proofs that deliver convergence *rates*, for example *log-linear convergence* which is the general lower bound in a black-box search scenario [44][78]. Convergence proofs without associated convergence rates are much easier to obtain [15][68], but they are of much less practical relevance. Such work will presumably be based on Markov Chain and drift analysis [6][13]. The previous results could be extended in several respects.

- investigating more general classes of objective functions, where also noisy objective function might be considered, see [45] for a first attempt.[15]
- investigating different types of algorithms, including derandomized variants of *evolution strategies*, see [32] for a first attempt, *estimation of distribution algorithms* [53] and the *cross entropy method* [67].

Such kind of results will have a stronger practical meaning than, for example, those obtained on *stochastic approximation* methods [75], because the latter rely on predefined gain sequences, which seem unrealistic in praxis. The methods with *adaptive* gain sequence are more interesting and more difficult to analyze. This kind of work would have a clear potential for becoming a milestone in stochastic search as well as optimization in general.

# 6 Conclusion

Variable metrics methods have a long and successful history in derivative-based numerical optimization. The probably most famous of these quasi-Newton methods is BFGS. More recently, derivative-free optimization has become popular, where derivatives are neither acquired nor estimated nor used. Also some of the most successful derivative-free optimization methods, like NEWUOA [62], are based on second order models and therefore acquire a variable metric.

Many evolutionary algorithms go one step beyond the derivative-free optimization paradigm. Not only do they disregard derivatives, but they do not even depend on specific function values. They are only based on the ordering of solutions and might be termed *function-value free*. This thesis has argued that variable metrics can be successfully assimilated in function-value-free optimization. The most prominent examples for such algorithms are EDAs [52] and the CMA-ES, while the latter often shows better scalability with the search space dimension.

More general, the thesis has provided a method, *adaptive encoding*, which allows to run generic, function-value free algorithms within a variable metric. It was proved that adaptive encoding recovers the CMA-ES, when applied to a simple evolution strategy. This result suggests that an efficient learning of a favorable variable metric has become available for many continuous search algorithms. While

[15] also: M. Jebalia, A. Auger and N. Hansen. Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. Accepted under minor revisions for *Algorithmica*, Springer.

an example application of adaptive encoding was given in [31], this concept needs yet to be widely explored with different algorithms.

Many difficult optimization problems exhibit strong dependencies between variables. Therefore, variable metrics methods are becoming indispensable in stochastic function-value free optimization at least in moderate dimensional search spaces.

## References


[1] S.I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] D. V. Arnold and H. G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1):135–159, 2003.

[3] D.V. Arnold. Weighted multirecombination evolution strategies. *Theoretical computer science*, 361(1):18–37, 2006.

[4] DV Arnold and H.G. Beyer. Performance analysis of evolutionary optimization with cumulative step length adaptation. *IEEE Transactions on Automatic Control*, 49(4):617–622, 2004.

[5] DV Arnold and R. Salomon. Evolutionary gradient search revisited. *IEEE Transactions on Evolutionary Computation*, 11(4):480–495, 2007.

[6] A. Auger. Convergence results for $(1,\lambda)$-SA-ES using the theory of $\varphi$-irreducible markov chains. *Theoretical Computer Science*, 334:35–69, 2005.

[7] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.

[8] Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 2009. accepted.

[9] T. Bartz-Beielstein and M. Preuss. Experimental research in evolutionary computation. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 3001–3020. ACM Press New York, NY, USA, 2007.

[10] Hans-Georg Beyer. Toward a theory of evolution strategies: On the benefit of sex - the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation*, 3(1):81–110, 1995.

[11] Hans-Georg Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heidelberg, 2001.

[12] Hans-Georg Beyer. Evolution strategies. *Scholarpedia*, page 15321, 2007.

[13] Alexis Bienvenüe and Olivier François. Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. *Theor. Comput. Sci.*, 306(1-3):269–289, 2003.

[14] D.Y. Cho, B.T. Zhang, and B. Labratory. Evolutionary Continuous Optimization by Distribution Estimation with Variational Bayesian Independent Component Analyzers Mixture Model. In *Parallel Problem Solving from Nature–PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004: Proceedings*. Springer, 2004.

[15] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, February 2002.

[16] W.C. Davidon. Variable Metric Method for Minimization. *SIAM J. Optim.*, 1:1, 1991.

[17] K. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[18] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary computation*, 10(4):371–395, 2002.

[19] LCW Dixon and GP Szego. The global optimization problem: an introduction. *Towards Global Optimization*, 2:1–15, 1978.

[20] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[21] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[22] A.E. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, 2002.

[23] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval schemata. *Foundations of Genetic Algorithms. 2*, pages 187–202, 1993.

[24] V. Feoktistov. *Differential Evolution: In Search of Solutions*. Optimization and Its Applications. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[25] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons Inc, 1966.

[26] Sylvain Gelly, Sylvie Ruette, and Olivier Teytaud. Comparison-based Algorithms are Robust and Randomized Algorithms are Anytime. *Evolutionary Computation Journal*, 15(4):411–434, 2007.

[27] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[28] D. Goldfarb. A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, 1970.

[29] N. Hansen. *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Eine Untersuchung zur entstochastisierten, koordinatensystemunabhängigen Adaptation der Mutationsverteilung*. Mensch und Buch Verlag, Berlin, 1998. ISBN 3-933346-29-0.

[30] N. Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[31] N. Hansen. Adaptive Encoding: How to Render Search Coordinate System Invariant. In G. Rudolph et al., editor, *Parallel Problem Solving from Nature (PPSN'08)*, LNCS, pages 205–214, 2008.

[32] N. Hansen. Toward a convergence proof for CMA-ES—and beyond. In D.V. Arnold, A.Auger, C.Witt, and J.E. Rowe, editors, *Theory of Evolutionary*

*Algorithms, Abstracts Collection*, number 08051 in Dagstuhl Seminar Abstracts Collection, Dagstuhl, Germany, 2008. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[33] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. *Parallel Problem Solving from Nature-PPSN VIII, LNCS*, 3242:282–291, 2004.

[34] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation. *Evolutionary Computation*, 11(1):1–18, 2003.

[35] N. Hansen, A.S.P. Niederberger, L. Guzzella, and P. Koumoutsakos. Evolutionary optimization of feedback controllers for thermoacoustic instabilities. In J.F. Morrison, D. M. Birch, and P. Lavoie, editors, *IUTAM Symposium on Flow Control and MEMS, Proceedings of the IUTAM Symposium held at the Royal Geographical Society, 19-22 September 2006, hosted by Imperial College, London, England*, volume 7 of *IUTAM Bookseries*. Springer Verlag, 2008.

[36] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317, 1996.

[37] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[38] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1995.

[39] T. Hastie and W. Stuetzle. Principle curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[40] W. Hock and K. Schittkowski. Test examples for nonlinear programming codes. *Journal of Optimization Theory and Applications*, 30(1):127–129, 1980.

[41] J.H. Holland. *Adaption in Natural and Artificial Systems.* PhD thesis, University of Michigan Press, 1975.

[42] C. Igel and M. Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):313–322, 2004.

[43] C. Igel and M. Toussaint. A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):313–322, 2004.

[44] J. Jägersküpper. Lower bounds for randomized direct search with isotropic sampling. *Operations Research Letters*, 36(3):327–332, 2008.

[45] M. Jebalia and A. Auger. On multiplicative noise models for stochastic search. In G. Rudolph, T. Jansen, S. Lucas, C. Polini, and N. Beume, editors, *Proceedings of Parallel Problem Solving from Nature (PPSN X)*, volume 5199 of *Lecture Notes in Computer Science*, pages 52–61. Springer Verlag, 2008.

[46] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):3–12, 2005.

[47] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–318, 2005.

[48] B. Kegl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(3):281–297, 2000.

[49] S. Kern, S. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms – A comparative review. *Natural Computing*, 3:77–112, 2004.

[50] J.N. Knight and M. Lunacek. Reducing the space-time complexity of the CMA-ES. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 658–665. ACM Press New York, NY, USA, 2007.

[51] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[52] P. Larrañaga. A review on estimation of distribution algorithms. In P. Larrañaga and J.A. Lozano, editors, *Estimation of distribution algorithms*, pages 80–90. Kluwer Academic Publishers, 2002.

[53] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[54] F.-L. Luo, R. Unbehagen, and A. Cichocki. A minor component analysis algorithm. *Neural Networks*, 10(2):291–297, 1997.

[55] R. Möller. A self-stabilizing learning rule for minor component analysis. *Int. J. Neural Syst.*, 14(1):1–8, 2004.

[56] J.J. Moré, B.S. Garbow, and K.E. Hillstrom. Testing Unconstrained Optimization Software. *ACM Transactions on Mathematical Software (TOMS)*, 7(1):17–41, 1981.

[57] E. Oja. Principal components, minor components, and linear networks. *Neural Networks*, 5(6):927–935, 1992.

[58] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. Step-size adaption based on non-local use of selection information. In *Parallel Problem Solving from Nature (PPSN III)*, pages 189–198. Springer-Verlag, 1994.

[59] J. Poland and A. Zell. Main Vector Adaptation: A CMA Variant with Linear Time and Space Complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1050–1055. Morgan Kaufmann, 2001.

[60] P. Pošík. Kernel principal components analysis as an efficient crossover operator in real-valued evolutionary algorithms. In *IEEE 4th International Conference on Intelligent Systems Design and Applications 2004*, pages 25–30, Piscataway, 2004. IEEE. ISBN 963-7154-29-9.

[61] P. Pošík. On the utility of linear transformations for population-based optimization algorithms. In *Preprints of the 16th World Congress of the International Federation of Automatic Control*, Prague, 2005. IFAC. CD-ROM.

[62] M.J.D. Powell. The NEWUOA software for unconstrained optimization without derivatives. *NONCONVEX OPTIMIZATION AND ITS APPLICATIONS*, 83:255, 2006.

[63] N.J. Radcliffe and P.D. Surry. Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective. *Lecture Notes in Computer Science, Springer Verlag, New York, NY*, 1000:275–291, 1995.

[64] I. Rechenberg. *Evolutionsstrategie '94*. Werkstatt Bionik und Evolutionstechnik. Frommann-Holzboog, Stuttgart, 1994.

[65] Ingo Rechenberg. *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. frommann-holzboog, Stuttgart, 1973.

[66] R. Ros and N. Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In G. Rudolph et al., editor, *Parallel Problem Solving from Nature (PPSN'08)*, LNCS, pages 296–305, 2008.

[67] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer, 2004.

[68] Günter Rudolph. Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26(3):375–390, 1997.

[69] T.P. Runarsson. Ordinal regression in evolutionary computation. In T. Runarsson, editor, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Proceedings*, volume 4193 of *Lecture notes in computational science*, pages 1048–1057. Springer, 2006.

[70] R. Salomon. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, 1998.

[71] B. Schoelkopf, A.J. Smola, and K.R. Mueller. Kernel Principal Component Analysis. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 583–588, 1997.

[72] M. Schumer and K. Steiglitz. Adaptive step size random search. *Automatic Control, IEEE Transactions on*, 13(3):270–276, 1968.

[73] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, 1994.

[74] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. John Wiley & Sons Inc., New York, 1995.

[75] J.C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853, 2000.

[76] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report Kanpur #2005005, Nanyang Technological University Singapore, and IIT Kanpur, India, May 2005. http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/Tech-Report-May-30-05.pdf.

[77] Thorsten Suttorp, Nikolaus Hansen, and Christian Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 72(2):167–197, 2009.

[78] O. Teytaud and S. Gelly. General lower bounds for evolutionary algorithms. In T. Runarsson et al., editors, *Parallel Problem Solving from Nature (PPSN'06)*, number 4193 in Lecture Notes in Computer Science, pages 21–31, Reykjavik, 2006.

[79] M. Welling, F. Agakov, and C. Williams. Extreme components analysis. In *NIPS 2003*, pages 137–144, 2003.

[80] D. Whitley. Functions as permutations: regarding no free lunch, walsh analysis and summary statistics. *Lecture notes in computer science*, pages 169–180, 2000.

[81] D. Whitley, S. Rana, J. Dzubera, and K.E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.

[82] D. Whitley and J. Rowe. Focused no free lunch theorems. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 811–818. ACM New York, NY, USA, 2008.

[83] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[84] Q. Zhang, NM Allinson, and H. Yin. Population optimization algorithm based on ICA. In *Combinations of Evolutionary Computation and Neural Networks, 2000 IEEE Symposium on*, pages 33–36, 2000.