Mach Learn (2009) 75: 167–197
DOI 10.1007/s10994-009-5102-1

Efficient covariance matrix update for variable metric evolution strategies

Thorsten Suttorp · Nikolaus Hansen · Christian Igel

Received: 11 September 2007 / Revised: 5 December 2008 / Accepted: 19 December 2008 /
Published online: 24 January 2009

Abstract Randomized direct search algorithms for continuous domains, such as evolution strategies, are basic tools in machine learning. They are especially needed when the gradient of an objective function (e.g., loss, energy, or reward function) cannot be computed or estimated efficiently. Application areas include supervised and reinforcement learning as well as model selection. These randomized search strategies often rely on normally distributed additive variations of candidate solutions. In order to efficiently search in non-separable and ill-conditioned landscapes the covariance matrix of the normal distribution must be adapted, amounting to a variable metric method. Consequently, covariance matrix adaptation (CMA) is considered state-of-the-art in evolution strategies. In order to sample the normal distribution, the adapted covariance matrix needs to be decomposed, requiring in general $\Theta(n^3)$ operations, where n is the search space dimension. We propose a new update mechanism which can replace a rank-one covariance matrix update and the computationally expensive decomposition of the covariance matrix. The newly developed update rule reduces the computational complexity of the rank-one covariance matrix adaptation to $\Theta(n^2)$ without resorting to outdated distributions. We derive new versions of the elitist covariance matrix adaptation evolution strategy (CMA-ES) and the multi-objective CMA-ES. These algorithms are equivalent to the original procedures except that the update step for the variable metric distribution scales better in the problem dimension. We also introduce a simplified variant of the non-elitist CMA-ES with the incremental covariance matrix update and investigate its performance. Apart from the reduced time-complexity of the distribution update, the algebraic computations involved in all new algorithms are simpler compared to the original

Editor: Risto Miikkulainen.

T. Suttorp · C. Igel (✉)
Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany
e-mail: christian.igel@neuroinformatik.rub.de

T. Suttorp
e-mail: thorsten.suttorp@neuroinformatik.rub.de

N. Hansen
Microsoft Research—INRIA Joint Centre & TAO, INRIA Saclay—Île-de-France,
Parc Orsay Université, 28 rue Jean Rostand, 91893 Orsay Cedex, France
e-mail: nikolaus.hansen@inria.fr

versions. The new update rule improves the performance of the CMA-ES for large scale machine learning problems in which the objective function can be evaluated fast.

Keywords Stochastic optimization · Variable metric algorithm · Multi-objective optimization · Evolutionary algorithm · Evolution strategy · Covariance matrix adaptation · Reinforcement learning

1 Introduction

Evolution strategies (ESs, Rechenberg 1973; Schwefel 1995; Beyer and Schwefel 2002) are randomized direct search algorithms. They are one of the major branches of evolutionary algorithms, a class of algorithms which draws inspiration from principles of neo-Darwinian evolution theory. Although ESs can be applied to various kinds of machine learning problems, the most elaborate variants are specialized for real-valued parameter spaces. Exemplary applications include supervised learning of feed-forward and recurrent neural networks, direct policy search in reinforcement learning, and model selection for kernel machines (e.g., Mandischer 2002; Igel et al. 2001; Schneider et al. 2004; Igel 2003; Friedrichs and Igel 2005; Kassahun and Sommer 2005; Pellecchia et al. 2005; Mersch et al. 2007; Siebel and Sommer 2007; Heidrich-Meisner and Igel 2008a, 2008b, 2008c; Glasmachers and Igel 2008, see below).

Evolution strategies and many other evolutionary algorithms for real-valued optimization, such as variants of evolutionary programming (Fogel 1995), rely on Gaussian random mutations. Candidate solutions are altered by adding random vectors drawn according to multi-variate zero-mean normal distributions. Adaptation of the covariance matrices of these random variations during optimization means learning and employing an appropriate metric for the search process. It is well-known that an appropriate *adaptation of the mutation strength* (step size adaptation) is virtually indispensable and assists the approach of an optimum with increasing precision (Schumer and Steiglitz 1968; Rechenberg 1973; Schwefel 1995). Additionally, the *adaptation of the shape* of the mutation distribution is appropriate. Adapting an arbitrary ellipsoidal shape (Schwefel 1995; Hansen et al. 1995; Hansen and Ostermeier 2001) amounts to a variable-metric approach in which a Mahalanobis metric is learned. Adapting the distribution shape can improve the search performance by orders of magnitude, especially for non-separable or badly scaled objective functions (Hansen and Ostermeier 2001; Beyer and Schwefel 2002; Kern et al. 2004). The covariance matrix adaptation ES (CMA-ES) implements both, adaptation of mutation strength and adaptation of distribution shape, explicitly, based on three major concepts: (a) the likelihood of successful steps is increased; (b) the correlation between successive steps in the past is exploited such that future successive steps tend to become uncorrelated (perpendicular); (c) invariance properties are sustained. The CMA-ES is regarded as one of the most powerful evolutionary algorithms for continuous optimization (Kern et al. 2004; Beyer 2007).

In order to sample a multi-variate normal distribution, the covariance matrix has to be factorized. The covariance matrix update and the sampling of the normal distribution require $\Theta(n^2)$ computations. Numerical routines for covariance matrix factorization require in general $\Theta(n^3)$ computations in n-dimensional search spaces. Consequently, on high dimensional objective functions that can be evaluated quickly, the factorization dominates the overall computational time. In order to reduce the computational burden on such objective functions to quadratic time, the factorization needs to be postponed until after $\Omega(n)$ iterations and consequently outdated distributions must be sampled.

In the following, we present a new update scheme for the covariances of Gaussian mutations in evolutionary algorithms that solves this problem. It operates directly on the factorization of the covariance matrix and reduces the computational complexity to $\Theta(n^2)$, the theoretical minimum for a matrix update. The update scheme also proves that it is always possible to update the covariance matrix and the distribution in each iteration with an asymptotic computational complexity of $\Theta(n^2)$. In addition, it simplifies the algebraic operations in the CMA-ES.

In the remainder of the introduction, we review performance evaluations and machine learning applications involving ESs with an emphasis on the CMA-ES. Then, we summarize previous work on reducing the time complexity of the covariance matrix update in ESs. In Sect. 2, we discuss Gaussian mutations and the adaptation of the mutation distribution in ESs. After that, we introduce our new update scheme. We incorporate the update scheme into the elitist $(1 + 1)$-CMA-ES and into the steady-state multi-objective CMA-ES (MO-CMA-ES, Igel et al. 2007a, 2007b). Then, we introduce a new non-elitist strategy, denoted as $(\mu/\mu_{\mathrm{w}}, \lambda)$-Cholesky-CMA-ES, which is a simplified version of the original non-elitist CMA-ES with rank-one updating. We empirically investigate the performance of the new algorithm in comparison with the original $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES with rank-one updating.

1.1 Performance assessment and machine learning applications of the CMA-ES

The CMA-ES is recognized as one of the most competitive evolutionary algorithms for real-valued optimization. Numerous performance evaluations and performance comparisons have been conducted across different suites of benchmark problems (Hansen and Ostermeier 2001; Hansen and Kern 2004; Kern et al. 2004; Auger and Hansen 2005a, 2005b; Whitley et al. 2006). The number of objective function evaluations in the optimization process scales, with few exceptions, between linearly and quadratically with the search space dimension n (Hansen and Ostermeier 2001; Hansen et al. 2003; Kern et al. 2004). Linear scaling must be regarded as a general lower bound (Jägersküpper 2007, 2008).

Arguably the most comprehensive performance comparison of CMA-ES with other bio-inspired, evolutionary and hybrid search methods for real parameter spaces took place in the *Session on Real-Parameter Optimization* at the 2005 *IEEE Congress on Evolutionary Computation*.[1] Performance results on 25 benchmark functions in 10, 30 and 50 dimensions were published for 11 algorithms (from 17 submissions), among them differential evolution, real-coded genetic algorithms, estimation of distribution algorithms, and hybrid particle swarm optimization. The $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES was applied in a restart setting, where the population size was increased by a factor of two before each restart (Auger and Hansen 2005b). The restarted CMA-ES was the clear winner of the competition. It outperformed the best competitors by solving the most functions. In addition, it solved them for the most part considerably faster (often by an order of magnitude) whenever a quantitative assessment was available. On both the subset of all unimodal functions and on the subset of all multi-modal functions, the CMA-ES achieved the best overall results.

Summarizing the performance assessments with reference to benchmark function properties, we find that only on essentially separable[2] objective functions CMA-ES can be significantly outperformed by other bio-inspired or hybrid algorithms. On essentially non-separable, ill-conditioned functions, CMA-ES generally outperforms them by orders of magnitude.

[1] For details see http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-05/CEC05.htm.

[2] Separable objective functions can be optimized coordinate-wise by n independent one-dimensional optimizations and therefore are not subject to the curse of dimensionality.

The CMA-ES has been successfully applied to a wide range of real-world problems (see the list by Hansen 2008). Most recently, Winter et al. (2008) conducted a comparison of the CMA-ES with gradient-based approaches—including a conjugate gradient algorithm and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method—in the domain of medical image registration, where the ES showed superior performance. In sum, Beyer (2007) observed that "CMA-ESs represent the state-of-the-art in evolutionary optimization in real-valued $\mathbb{R}^n$ search spaces".

Using ESs for parameter optimization is particularly advisable in the presence of noise (Arnold 2002; Arnold and Beyer 2003), where standard gradient-based approaches fail. Arnold and Beyer (2003) compared ESs to several other direct search methods in a noisy environment and the results support the evolutionary approach. In machine learning, noisy objective functions are common. For example, they arise when the reward to be optimized is determined by a finite number of episodes (or roll-outs), the standard scenario in reinforcement learning. They also arise in supervised learning and model selection when the learning system performance is assessed through randomly generated samples of data.

Evolution strategies and in particular the CMA-ES have been successfully applied in various fields of machine learning. The following examples highlight this success, emphasizing our own work.

1.1.1 Supervised learning

The CMA-ES is well-suited for supervised learning in scenarios where the objective function is highly noisy and/or multi-modal, in particular if gradient information cannot be obtained efficiently. Although Mandischer (2002) considered ESs for feed-forward neural network training, the strength of this approach becomes more apparent when training recurrent systems. For example, Igel et al. (2001) and Schneider et al. (2004) compared the CMA-ES with the BFGS based on analytically derived gradients for learning the parameters of neural fields, and the CMA-ES clearly outperformed the quasi-Newton method.

Mierswa (2006, 2007) investigated ESs for support vector machine (SVM) training, in particular in the case of non positive semi-definite kernel functions. In his recent work, he applied a multi-objective ES to SVM training, directly addressing the trade-off between model complexity and empirical risk (Mierswa 2007). This approach does not require an *a priori* chosen regularization parameter and can be viewed as a combination of training and model selection (see below).

1.1.2 Reinforcement learning

Evolutionary algorithms are reinforcement learning (RL) methods in the definition of Sutton and Barto (1998). They are typically used for direct policy search. Igel (2003) brought forward the CMA-ES in the framework of RL. He found that the CMA-ES with rank-one updating outperforms alternative evolutionary RL approaches on variants of the pole balancing benchmark in fully and partially observable environments. In a more recent study by Gomez et al. (2006), these results were compared to 8–12 (depending on the task) other RL algorithms including value-function and policy gradient approaches. On the four test problems where the CMA-ES was considered, it ranked first, second (twice), and third.

Pellecchia et al. (2005) applied the CMA-ES to model human car driving behavior using neural attractor dynamics to represent the policies. Siebel and Sommer (2007) and Kassahun and Sommer (2005) employed the CMA-ES for RL in robotics. They combined the CMA-ES with evolutionary topology optimization to evolve artificial neutral networks.

Recently, Heidrich-Meisner and Igel (2008a, 2008b, 2008c) performed a systematic comparison between the CMA-ES and policy gradient methods with variable metrics. They discuss similarities and differences between these related approaches. Preliminary experiments—where the methods operated on the same class of policies and the same benchmark problems—indicate that the CMA-ES learns at least as quickly while being much more robust regarding the choice of hyperparameters and initial policies.

1.1.3 Model selection

Real-valued evolutionary algorithms are often used for model selection of SVMs (see e.g., Runarsson and Sigurdsson 2004; Friedrichs and Igel 2005; Igel 2005; Suttorp and Igel 2006; Mersch et al. 2007; Glasmachers and Igel 2008). Compared to gradient-based optimization, they do not require the class of kernel functions to have a differentiable structure, they are applicable even if the score function for assessing parameter performance is not differentiable, and they are less prone to converge to suboptimal solutions. Unlike nested grid-search, they also allow for adapting multiple hyperparameters. The CMA-ES was successfully applied to adapt general Gaussian kernels (Friedrichs and Igel 2005; Glasmachers and Igel 2008) as well as string kernels (Mersch et al. 2007). Successful multi-objective model selection for SVMs using real-valued evolutionary algorithms is described by Igel (2005) and Suttorp and Igel (2006).

1.2 Previous work in complexity reduction of the covariance matrix update in ESs

Several heuristics have been proposed to reduce the time complexity of the covariance matrix update. Already in the first article on CMA-ES, Hansen and Ostermeier (1996) suggested to postpone the update of the covariance matrix for n iterations. This, however, leads to sampling outdated distributions. Nevertheless, in practice, the update is only performed every τ generations. For $\tau = o(n)$ the computational complexity is still $\omega(n^2)$ if the algorithm relies on an eigenvalue decomposition, while $\tau = \omega(n)$ is not advisable. Only when $\tau = \Theta(n)$ is the computational complexity $\Theta(n^2)$, the same as for the update scheme developed in this article. This approach has three drawbacks. First, sampling outdated search distributions reduces the search performance in terms of progress achieved per objective function evaluation and therefore increases the overall running time. This is shown in Fig. 1. Second, the decomposition leads to peak computations every τ generations. Third, an internal parameter, the update frequency, needs to be chosen properly.

Igel et al. (2006) derived a distribution update in quadratic time under the constraint that only the variations from the previous iteration (i.e., from a single generation) are considered. This update rule was applied within simplified variants of the elitist CMA-ES for single- and multi-objective optimization (Igel et al. 2006, 2007b). The simplification was necessary, because the covariance update in the CMA-ES is originally based on a weighted average of previously taken steps (the *evolution path*, see below) and not just on the last iteration. Depending on the optimization problem, this simplification can lead to a significant performance decrease both in the single- and multi-objective case as described by Igel et al. (2006, 2007b) and demonstrated in Sect. 4.1 below (Fig. 4).

Knight and Lunacek (2007) proposed to adapt the covariance only in an l-dimensional subspace, where $l < n$ is chosen *a priori*. This restriction allows for an update step in $O(l^2 n)$. Knight and Lunacek (2007) showed that the performance decreases significantly if the necessary dimensionality is underestimated. On a generic test function (f_{elli} from below, where $n = 30$) the variant slows down by more than a factor of ten, even with $l = \frac{n}{2}$ (Knight and Lunacek 2007, erratum).

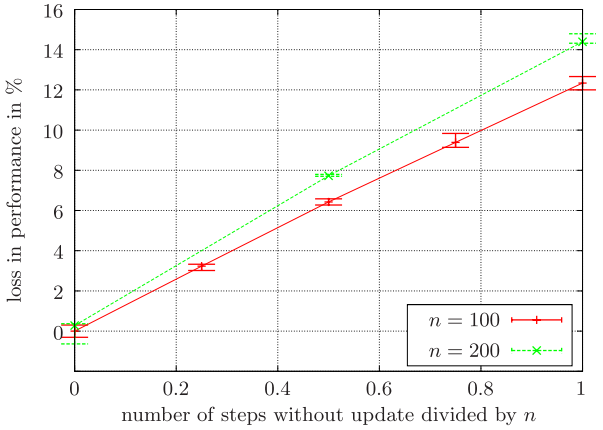

Fig. 1 Additionally needed function evaluations (in percent) to reach 10^{-30} on f_{elli} (see Table 2) when outdated distributions are sampled. The results refer to the default $(\mu/\mu_{\text{w}}, \lambda)$-CMA-ES without rank-$\mu$ update of the covariance matrix (Hansen and Ostermeier 2001). Curves for the standard variant with rank-μ update look virtually the same. The *abscissa* shows the number of steps the update is postponed divided by problem dimensionality. The *lower curve* depicts the median, the error bars the 10th and 90th percentile, for $n = 100$ dimensions and 11 trials for each case. The *upper line* shows the results for $n = 200$ dimensions from 3 trials. Waiting for n iteration steps leads to a loss between 12 and 14 percent

Algorithms that can learn covariances in linear time were proposed previously (Ostermeier 1992; Hansen et al. 1995; Poland and Zell 2001). Not surprisingly, these variants also suffer from a limited model complexity because the number of updated parameters is linear in n. They perform comparatively poorly whenever a full covariance matrix is required from the underlying search space landscape.

The algorithms presented in the following do not suffer from any of these limitations. They learn arbitrary covariance matrices, they do not sample outdated distributions, and they utilize an evolution path.

1.3 Limitations

Any search and optimization algorithm must exploit properties of the objective function, and the question arises on which classes of problems the CMA-ES will perform badly (Igel and Toussaint 2003). First, the CMA-ES relies on neighborhood, assuming a correlation between the distance of solutions and the dissimilarity in their objective function values (Jones and Forrest 1995). The distance measure is based on the variable metric. When the neighborhood information is useless or misleading, CMA-ES will perform poorly (like most search algorithms). Second, invariance properties are a major design criterion for the CMA-ES (see below). Invariance, however, always coincides with lack of specialization. For example, a class of optimization problems might share a common coordinate system that is optimal for solving all instances. On such problems, algorithms that *a priori* utilize the distinguished coordinate system are potentially better than coordinate system invariant algorithms. Indeed, CMA-ES can be outperformed on separable problems, where the given coordinate system is highly distinguished. Similarly, it can be beneficial to sacrifice invariance under order preserving transformation of the function value (which is due to the rank-based selection in ESs). For example, on *noise-free* quadratic functions, quasi-Newton methods are typically an order of magnitude faster than CMA-ES, because they exploit the function *values* explicitly.

Finally, the internal computations of CMA-ES scale at least with n^2. On high dimensional quickly to evaluate objective functions, linearly scaling algorithms can be advantageous, even though they lack the ability to learn all mutual dependencies between variables (Ros and Hansen 2008). It is the objective of this work to reach the n^2 lower bound without any restrictions on the dependencies that can be learned and without resorting to outdated distributions.

2 Gaussian mutations in evolution strategies

We consider ESs for real-valued optimization (Rechenberg 1973; Schwefel 1995; Beyer and Schwefel 2002; Kern et al. 2004), which are quite well understood theoretically (cf. Beyer 2001; Auger 2005; Jägersküpper 2006, 2008). Let the optimization problem be defined by an objective function $f : \mathbb{R}^n \to \mathcal{Y}$ to be minimized, where n denotes the dimensionality of the search space (space of candidate solutions, decision space) and $\mathcal{Y}$ the space of cost values. Evolution strategies are random search methods, which iteratively sample a set of candidate solutions from a probability distribution over the search space, evaluate these points using f, and construct a new probability distribution over the search space based on the gathered information. In typical ESs, this search distribution is parameterized by a set of candidate solutions, the *parent population*, and by parameters of the variation operators that are used to create new candidate solutions (the *offspring*) from the parent population.

In a canonical ES the objective vector $\boldsymbol{x}_i^{(g+1)} \in \mathbb{R}^n$ of the ith offspring at generation g is created by

$$\boldsymbol{x}_i^{(g+1)} \leftarrow \underbrace{\boldsymbol{c}_i^{(g)}}_{\text{recombination}} + \underbrace{\boldsymbol{v}_i^{(g)}}_{\text{mutation}}.$$

The vector $\boldsymbol{c}_i^{(g)}$ depends on the *recombination scheme*. For example, in case of *weighted global intermediate recombination* $\boldsymbol{c}_i^{(g)}$ is the weighted center of mass of the objective vectors in the current parent population. If no recombination is used, $\boldsymbol{c}_i^{(g)}$ is simply the objective vector of one of the parents. The mutation $\boldsymbol{v}_i^{(g)}$ is a realization of an n-dimensional random vector distributed according to a zero-mean Gaussian distribution with covariance matrix $\boldsymbol{C}_i^{(g)}$, that is,

$$\boldsymbol{v}_i^{(g)} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{C}_i^{(g)}\right).$$

Sampling the mutation distribution to generate $\boldsymbol{v}_i^{(g)}$ is usually conducted in two steps. First, the standard normal distribution is sampled to generate a realization of an n-dimensional normally distributed random vector $\boldsymbol{z}_i^{(g)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with unit covariance matrix and zero mean. Second, this random vector is rotated and scaled by a linear transformation $\boldsymbol{A}_i^{(g)} \in \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{A}_i^{(g)}\boldsymbol{z}_i^{(g)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_i^{(g)}) \quad \text{for } \boldsymbol{z}_i^{(g)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}).$$

To obtain $\boldsymbol{A}_i^{(g)}$ and to sample a general multivariate normal distribution, the covariance matrix $\boldsymbol{C}_i^{(g)}$ is decomposed into Cholesky factors

$$\boldsymbol{C}_i^{(g)} = \boldsymbol{A}_i^{(g)}\boldsymbol{A}_i^{(g)^{\mathrm{T}}}.$$

Every symmetric nonnegative definite matrix, such as covariance matrices, has real-valued Cholesky factors. We do not require the matrices $\boldsymbol{A}_i^{(g)}$ to be of a special form (Grewal and

Andrews 1993), for example triangular. The factorization is in general not unique (by imposing certain constraints, e.g., $\boldsymbol{A}_i^{(g)}$ lower triangular with nonnegative diagonal elements, a unique decomposition can be defined). Numerical routines for computing Cholesky factors by a triangular Cholesky, eigenvalue, or singular value decomposition algorithm for a general covariance matrix require $\Theta(n^3)$ steps.

One of the decisive features of ESs is that the covariance matrices are subject to adaptation. The general strategy is to alter the covariance matrices such that steps promising a large fitness gain are sampled more often. There are typically two ways how the adaptation of the matrices is realized. First, the covariance matrix or its Cholesky factors can be parameterized, and these parameters can then be adapted. Either the parameterization or the adaptation rule has to ensure that the resulting matrices remain positive definite. In self-adaptive ESs the parameterization guarantees positive definiteness (e.g., see Rudolph 1992), and the parameters are changed by mutation. Here we consider a second, more efficient way, where the covariance matrix is directly altered by additive updates of the form

$$\boldsymbol{C}^{(g+1)} = \alpha \boldsymbol{C}^{(g)} + \beta \boldsymbol{V}^{(g)}.$$

The matrix $\boldsymbol{V}^{(g)} \in \mathbb{R}^{n \times n}$ is assumed to be positive definite and $\alpha, \beta \in \mathbb{R}^+$ are weighting factors. Let $\boldsymbol{v}^{(g)} \in \mathbb{R}$ be a step in the search space promising large fitness gain. To increase the probability that $\boldsymbol{v}^{(g)}$ is sampled in the next iteration, the rank-one update

$$\boldsymbol{C}^{(g+1)} = \alpha \boldsymbol{C}^{(g)} + \beta \boldsymbol{v}^{(g)} \boldsymbol{v}^{(g)\mathrm{T}} \tag{1}$$

is appropriate. This update rule shifts the mutation distribution towards the line distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{v}^{(g)} \boldsymbol{v}^{(g)\mathrm{T}})$, which is the Gaussian distribution with the highest probability to generate $\boldsymbol{v}^{(g)}$ among all normal distributions with zero mean. After the update, the new covariance matrix has to be decomposed into Cholesky factors in order to sample the distribution. If the covariance matrix updates occur frequently in the ES the time needed for the factorization can dominate the computation time of the ES even for a moderate number of dimensions n.

3 Efficient covariance matrix update

In general, each factorization of a covariance matrix requires $\Theta(n^3)$ operations. Thus, in ESs with additive covariance matrix adaptation (e.g., using (1)) the Cholesky factorization of the covariance matrix is the computationally dominating factor apart from the fitness function evaluations. Therefore, we propose not to factorize the covariance matrix, but to use an incremental rank-one update rule for the Cholesky factorization. The idea is never to compute the covariance matrix explicitly, but to operate on Cholesky factors only. We consider Cholesky factors that are general $n \times n$-matrices, and give a general update rule for ESs. The proposed technique makes use of efficient rank-one matrix updates (Hager 1989), which are for example frequently used in the domain of Kalman filtering (Grewal and Andrews 1993).

First, we derive an intermediate update rule for the special case where $\boldsymbol{v}^{(g)} = \boldsymbol{A}^{(g)} \boldsymbol{z}^{(g)}$ and $\boldsymbol{z}^{(g)}$ is given[3] (Igel et al. 2006). Under this assumption we can rewrite the rank-one update

[3]The vector $\boldsymbol{z}^{(g)}$ is, for example, the realization of a standard normally distributed random variable and an individual has been mutated by adding a vector proportional to $\boldsymbol{v}^{(g)} = \boldsymbol{A}^{(g)} \boldsymbol{z}^{(g)}$.

of the covariance matrix (1) as

$$\boldsymbol{C}^{(g+1)} = \alpha \boldsymbol{C}^{(g)} + \beta \boldsymbol{A}^{(g)} \boldsymbol{z}^{(g)} \left[\boldsymbol{A}^{(g)} \boldsymbol{z}^{(g)}\right]^{\mathrm{T}}.$$

Our goal is to turn this update for $\boldsymbol{C}^{(g)}$ into an update for $\boldsymbol{A}^{(g)}$. To achieve this, we derive the following lemma.

Lemma 1 (Igel et al. 2006) *Let $\boldsymbol{u} \in \mathbb{R}^n$ be an arbitrary vector. Then, the matrix $\boldsymbol{I} + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}$ can be decomposed into*

$$\boldsymbol{I} + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} = (\boldsymbol{I} + \varsigma \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}})(\boldsymbol{I} + \varsigma \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}})$$

with $\varsigma = \frac{1}{2}$ for $\boldsymbol{u} = \boldsymbol{0}$ and $\varsigma = \frac{1}{\|\boldsymbol{u}\|^2}(\sqrt{1 + \|\boldsymbol{u}\|^2} - 1)$ otherwise.

Proof We have

$$\begin{aligned}(\boldsymbol{I} + \varsigma \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}})(\boldsymbol{I} + \varsigma \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}) &= \boldsymbol{I} + 2\varsigma \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} + \varsigma^2 \|\boldsymbol{u}\|^2 \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} \\ &= \boldsymbol{I} + \left(2\varsigma + \varsigma^2 \|\boldsymbol{u}\|^2\right) \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} \\ &= \boldsymbol{I} + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}.\end{aligned}$$

The last step holds because $(2\varsigma + \varsigma^2 \|\boldsymbol{u}\|^2) = 1$ by definition of ς. □

This result allows us to prove the following theorem, which shows how to realize an incremental update of the Cholesky factors given already appropriately factorized update vectors.

Theorem 1 (Igel et al. 2006) *Let $\boldsymbol{C}_t \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix with Cholesky factorization $\boldsymbol{C}_t = \boldsymbol{A}_t \boldsymbol{A}_t^{\mathrm{T}}$. Assume further that $\boldsymbol{C}_t$ is updated using*

$$\boldsymbol{C}_{t+1} = \alpha \boldsymbol{C}_t + \beta \boldsymbol{v}_t \boldsymbol{v}_t^{\mathrm{T}},$$

where $\boldsymbol{v}_t \in \mathbb{R}^n$ is given in the decomposed form $\boldsymbol{v}_t = \boldsymbol{A}_t \boldsymbol{z}_t$, and $\alpha, \beta \in \mathbb{R}^+$. For $\boldsymbol{z}_t \neq \boldsymbol{0}$ a Cholesky factor of the matrix $\boldsymbol{C}_{t+1}$ can be computed by

$$\boldsymbol{A}_{t+1} = \sqrt{\alpha} \boldsymbol{A}_t + \frac{\sqrt{\alpha}}{\|\boldsymbol{z}_t\|^2} \left(\sqrt{1 + \frac{\beta}{\alpha} \|\boldsymbol{z}_t\|^2} - 1 \right) [\boldsymbol{A}_t \boldsymbol{z}_t] \boldsymbol{z}_t^{\mathrm{T}}, \tag{2}$$

for $\boldsymbol{z}_t = \boldsymbol{0}$ we have $\boldsymbol{A}_{t+1} = \sqrt{\alpha} \boldsymbol{A}_t$.

Proof For $\boldsymbol{z}_t \neq \boldsymbol{0}$ it holds

$$\begin{aligned}\boldsymbol{C}_{t+1} &= \alpha \boldsymbol{C}_t + \beta \boldsymbol{v}_t \boldsymbol{v}_t^{\mathrm{T}} \\ &= \alpha \boldsymbol{A}_t \boldsymbol{A}_t^{\mathrm{T}} + \beta \boldsymbol{A}_t \boldsymbol{z}_t \boldsymbol{z}_t^{\mathrm{T}} \boldsymbol{A}_t^{\mathrm{T}} \\ &= \alpha \boldsymbol{A}_t \left(\boldsymbol{I} + \boldsymbol{z}_t \frac{\beta}{\alpha} \boldsymbol{z}_t^{\mathrm{T}} \right) \boldsymbol{A}_t^{\mathrm{T}} \\ &= \sqrt{\alpha} \boldsymbol{A}_t \left(\boldsymbol{I} + \varsigma \boldsymbol{z}_t \frac{\beta}{\alpha} \boldsymbol{z}_t^{\mathrm{T}} \right) \left(\boldsymbol{I} + \varsigma \boldsymbol{z}_t \frac{\beta}{\alpha} \boldsymbol{z}_t^{\mathrm{T}} \right) \boldsymbol{A}_t^{\mathrm{T}} \sqrt{\alpha},\end{aligned}$$

where $\varsigma = \frac{\alpha}{\beta}\frac{1}{\|z_t\|^2}\left(\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}-1\right)$. For the factorization in the last equality we used Lemma 1 with $\boldsymbol{u} = \sqrt{\frac{\beta}{\alpha}}z_t$. It directly follows that

$$\begin{aligned}
\boldsymbol{A}_{t+1} &= \sqrt{\alpha}\boldsymbol{A}_t\left(\boldsymbol{I}+\varsigma z_t\frac{\beta}{\alpha}z_t^{\mathrm{T}}\right)\\
&= \sqrt{\alpha}\boldsymbol{A}_t+\frac{\beta}{\sqrt{\alpha}}\varsigma\boldsymbol{A}_t z_t z_t^{\mathrm{T}}\\
&= \sqrt{\alpha}\boldsymbol{A}_t+\frac{\sqrt{\alpha}}{\|z_t\|^2}\left(\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}-1\right)[\boldsymbol{A}_t z_t]z_t^{\mathrm{T}}
\end{aligned}$$

computes the update of the Cholesky factor. The proof for the case $z_t = \mathbf{0}$ is obvious. □

The square brackets in the last equation indicate the order of computation, showing how to achieve a time complexity of $\Theta(n^2)$. The numerical stability of direct updates of Cholesky factors is likely to be better than updates requiring subsequent decompositions (e.g., see the discussion in Grewal and Andrews 1993, Chap. 6).

Now we derive, as a new extension to the previously derived results, a corresponding update for the inverse of the Cholesky factors, which can then be used to bring arbitrary updates $\boldsymbol{v}_t\boldsymbol{v}_t^{\mathrm{T}}$ in the form $\boldsymbol{A}_t z_t[\boldsymbol{A}_t z_t]^{\mathrm{T}}$ suitable for Theorem 1.

Theorem 2 *Let $\boldsymbol{A}_t, \boldsymbol{A}_{t+1} \in \mathbb{R}^{n\times n}$, $z_t \in \mathbb{R}^n$, and $\alpha, \beta \in \mathbb{R}^+$ be given such that (2) of Theorem 1 holds. Let $\boldsymbol{A}_t^{-1}$ be the inverse of $\boldsymbol{A}_t$. Then the inverse of $\boldsymbol{A}_{t+1}$ can be computed by*

$$\boldsymbol{A}_{t+1}^{-1} = \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_t^{-1}-\frac{1}{\sqrt{\alpha}\|z_t\|^2}\left(1-\frac{1}{\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}}\right)z_t\left[z_t^{\mathrm{T}}\boldsymbol{A}_t^{-1}\right] \tag{3}$$

for $z_t \neq \mathbf{0}$ and by $\boldsymbol{A}_{t+1}^{-1} = \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_t^{-1}$ for $z_t = \mathbf{0}$.

Proof Let $z_t \neq \mathbf{0}$ and define $a := \sqrt{\alpha}$ and $b := \frac{\sqrt{\alpha}}{\|z_t\|^2}\left(\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}-1\right)$. With these definitions the inverse of $\boldsymbol{A}_{t+1}$ can be calculated as follows:

$$\begin{aligned}
\boldsymbol{A}_{t+1}^{-1} &= \left(a\boldsymbol{A}_t+b\boldsymbol{A}_t z_t z_t^{\mathrm{T}}\right)^{-1}\\
&= \frac{1}{a}\boldsymbol{A}_t^{-1}-\frac{\left(\frac{1}{a}\boldsymbol{A}_t^{-1}\right)\left(b\boldsymbol{A}_t z_t z_t^{\mathrm{T}}\right)\left(\frac{1}{a}\boldsymbol{A}_t^{-1}\right)}{1+z_t^{\mathrm{T}}\left(\frac{1}{a}\boldsymbol{A}_t^{-1}\right)\left(b\boldsymbol{A}_t z_t\right)}\\
&= \frac{1}{a}\boldsymbol{A}_t^{-1}-\frac{b/a^2}{1+b/a\cdot\|z_t\|^2}z_t z_t^{\mathrm{T}}\boldsymbol{A}_t^{-1}\\
&= \frac{1}{a}\boldsymbol{A}_t^{-1}-\frac{\sqrt{\alpha}\left(\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}-1\right)}{\alpha\|z_t\|^2}\cdot\frac{z_t z_t^{\mathrm{T}}\boldsymbol{A}_t^{-1}}{\left(1+\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}-1\right)}\\
&= \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_t^{-1}-\frac{1}{\sqrt{\alpha}\|z_t\|^2}\left(1-\frac{1}{\sqrt{1+\frac{\beta}{\alpha}\|z_t\|^2}}\right)z_t z_t^{\mathrm{T}}\boldsymbol{A}_t^{-1}.
\end{aligned}$$

The first equality follows from the definition of the update step for the Cholesky factor given by Theorem 1. For the second equality the Sherman-Morrison formula $(\boldsymbol{B} + \boldsymbol{x}\,\boldsymbol{y}^{\mathrm{T}})^{-1} = \boldsymbol{B}^{-1} - \frac{\boldsymbol{B}^{-1}\boldsymbol{x}\,\boldsymbol{y}^{\mathrm{T}}\boldsymbol{B}^{-1}}{1+\boldsymbol{y}^{\mathrm{T}}\boldsymbol{B}^{-1}\boldsymbol{x}}$ is used with $\boldsymbol{B} = a\boldsymbol{A}_t$, $\boldsymbol{x} = b\boldsymbol{A}_t\boldsymbol{z}_t$ and $\boldsymbol{y} = \boldsymbol{z}_t$ (Hager 1989). Note, that if $\boldsymbol{A}_0$ is invertible so is $\boldsymbol{A}_t$ for all $t \in \mathbb{N}_0$. The proof for the case $\boldsymbol{z}_t = \boldsymbol{0}$ is obvious. □

Now we can combine the theorems to derive our main result:

Corollary 1 *For covariance matrix updates of the form*

$$\boldsymbol{C}^{(g+1)} = \alpha\boldsymbol{C}^{(g)} + \beta\boldsymbol{v}^{(g)}\boldsymbol{v}^{(g)\mathrm{T}}$$

the Cholesky factors of $\boldsymbol{C}^{(g+1)}$ *and their inverses can be computed in* $\Theta(n^2)$ *time given the decomposition* $\boldsymbol{C}^{(g)} = \boldsymbol{A}^{(g)}\boldsymbol{A}^{(g)\mathrm{T}}$ *and* $\boldsymbol{A}^{(g)^{-1}}$. *Let* $\boldsymbol{z}^{(g)} = \boldsymbol{A}^{(g)^{-1}}\boldsymbol{v}^{(g)}$. *For* $\boldsymbol{v}^{(g)} \neq \boldsymbol{0}$ *we have*

$$\boldsymbol{A}^{(g+1)^{-1}} = \frac{1}{\sqrt{\alpha}}\boldsymbol{A}^{(g)^{-1}} - \frac{1}{\sqrt{\alpha}\|\boldsymbol{z}^{(g)}\|^2}\left(1 - \frac{1}{\sqrt{1+\frac{\beta}{\alpha}\|\boldsymbol{z}^{(g)}\|^2}}\right) \times \boldsymbol{z}^{(g)}\left[\boldsymbol{z}^{(g)\mathrm{T}}\boldsymbol{A}^{(g)^{-1}}\right] \tag{4}$$

and

$$\boldsymbol{A}^{(g+1)} = \sqrt{\alpha}\boldsymbol{A}^{(g)} + \frac{\sqrt{\alpha}}{\|\boldsymbol{z}^{(g)}\|^2}\cdot\left(\sqrt{1+\frac{\beta}{\alpha}\|\boldsymbol{z}^{(g)}\|^2} - 1\right)\boldsymbol{v}^{(g)}\boldsymbol{z}^{(g)\mathrm{T}}. \tag{5}$$

For $\boldsymbol{v}^{(g)} = \boldsymbol{0}$, *we have* $\boldsymbol{A}^{(g+1)^{-1}} = \frac{1}{\sqrt{\alpha}}\boldsymbol{A}^{(g)^{-1}}$ *and* $\boldsymbol{A}^{(g+1)} = \sqrt{\alpha}\boldsymbol{A}^{(g)}$.

Proof Theorem 1 can be applied for arbitrary vectors $\boldsymbol{v}^{(g)}$ because the inverse of the Cholesky factor is known. In this case, we can compute $\boldsymbol{z}^{(g)} = \boldsymbol{A}^{(g)^{-1}}\boldsymbol{v}^{(g)}$ and decompose $\boldsymbol{v}^{(g)}$ into $\boldsymbol{v}^{(g)} = \boldsymbol{A}^{(g)}(\boldsymbol{A}^{(g)^{-1}}\boldsymbol{v}^{(g)}) = \boldsymbol{A}^{(g)}\boldsymbol{z}^{(g)}$ with computational complexity $\Theta(n^2)$. The update of the Cholesky factor using this decomposition and Theorem 1 scales also with $\Theta(n^2)$. The required inverse of the Cholesky factor $\boldsymbol{A}^{(g)^{-1}}$ can be recursively updated using Theorem 2 in $\Theta(n^2)$ steps. □

Note that in applications of Corollary 1, many of the terms in the update equations are already available and need not to be re-computed. For an analogous rank-μ update[4] (e.g., see Hansen et al. 2003; Hansen and Kern 2004) the time complexity can be reduced accordingly to $\Theta(\mu n^2)$, where $\mu < n$.

[4]Rank-μ updates have the general form $\boldsymbol{C}^{(g+1)} = \alpha\boldsymbol{C}^{(g)} + \sum_{i=1}^{\mu}\beta_i\boldsymbol{v}_i^{(g)}\boldsymbol{v}_i^{(g)\mathrm{T}}$ with $\boldsymbol{v}_i^{(g)} \in \mathbb{R}^n$ and $\beta_i > 0$ for $1 \leq i \leq \mu$. The $n \times n$ matrix $\boldsymbol{V}^{(g)} = \sum_{i=1}^{\mu}\beta_i\boldsymbol{v}_i^{(g)}\boldsymbol{v}_i^{(g)\mathrm{T}}$ has rank μ if the vectors are linearly independent. The update can be realized iteratively, $\boldsymbol{C}^{(g+1)} = [[\alpha\boldsymbol{C}^{(g)} + \beta_1\boldsymbol{v}_1^{(g)}\boldsymbol{v}_1^{(g)\mathrm{T}}] + \beta_2\boldsymbol{v}_2^{(g)}\boldsymbol{v}_2^{(g)\mathrm{T}}\ldots]$. In practice, the rank-$\mu$ update is also used in the CMA-ES, where it is particularly useful with large offspring populations. Then, not only the weighted population mean, but also individual steps are considered in the update of the covariance matrix.

4 Applications

In this section, we show how the new covariance matrix update can be incorporated into state-of-the-art ESs. We consider recent variants of the covariance matrix adaptation ES (CMA-ES). The CMA-ES is our method of choice for real-valued evolutionary optimization for several reasons. First, the method follows appealing design principles (e.g., see Hansen 2006), in particular it puts strong emphasis on invariance properties. Second, the method is quasi-parameter free (Auger and Hansen 2005b). Third, as explicated above, the method performed well in benchmark scenarios (see Auger and Hansen 2005b) and, last but not least, also in many real-world applications (Hansen 2008).

The key idea of all CMA-ES algorithms is to alter the mutation distribution such that the probability is increased, to reproduce steps in search space that led to the actual population (i.e., produced offspring that were selected). Using a covariance matrix update enables the algorithm to detect correlations between the objective variables and to become invariant under transformations of the search space. The algorithms implement several important concepts. The first one is known as *derandomization*: the adaptation of step size, variance, and covariance parameters depends *deterministically* on the realized selected steps in search space—in contrast to mutative *self-adaptation* (Rechenberg 1973; Schwefel 1995), where the dependence is stochastic. The second principle is *cumulation*: the trick of taking the search path of the population over the past generations into account (evolution path), where the influence of previous steps decays exponentially. Further, all CMA-ES variants decouple the update of a global step size and a covariance matrix, that is, the mutation distribution is parameterized by

$$\boldsymbol{v}_i^{(g)} \sim \sigma^{(g)} \mathcal{N}\left(\mathbf{0}, \boldsymbol{C}_i^{(g)}\right) = \sigma^{(g)} \boldsymbol{A}_i^{(g)} \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right). \tag{6}$$

The Cholesky factor $\boldsymbol{A}$ determines the *shape* of the distribution. Its update is based on the idea to reinforce successful directions. The update of step size σ, taking place on a considerably shorter time scale, is supposed to ensure nearly maximal progress for a given distribution shape.

The update of the mutation distribution is conducted such that the CMA-ES is invariant under transformations of the optimization problem, in particular invariant under rotation and translation of the search space (of course, apart from the initialization). If the mutation distribution is initialized accordingly, any affine transformation of the search space does not affect the performance of the CMA-ES. The single-objective CMA-ES is invariant under order-preserving transformations of the fitness function value. The multi-objective CMA-ES is invariant under order-preserving affine transformations $f_i \mapsto a_i f_i + b_i$ with $a_i \in \mathbb{R}^+$ and $b_i \in \mathbb{R}$ if the reference point for computing the hypervolume ($\boldsymbol{x}_{\text{ref}}$, see Sect. 4.2.1) is transformed accordingly or can be neglected, as it is the case in the standard MO-CMA-ES implementation.[5]

Evolution strategies use rank-based selection. Given μ parents and λ offspring, a (μ, λ) selection strategy chooses the μ best of the offspring to form the next parent population. A $(\mu+\lambda)$ strategy chooses the μ best individuals from the union of parents and offspring. The latter is an elitist selection method, because the best solution found so far is always a member of the population.

In the following, we present the elitist CMA-ES and the multi-objective MO-CMA-ES using the efficient learning of the metric. For completeness, we concisely explain the

[5] We thank the anonymous reviewer for pointing out the extended invariance properties of the MO-CMA-ES.

algorithms based on the original publications. Then, we introduce a new variant of the non-elitist CMA-ES compatible with the incremental covariance matrix update. For notational convenience, we do not consider the case of the mutation vector being zero (i.e., $z = \mathbf{0}$) in the description of the algorithms, a case which occurs with probability zero.

4.1 Elitist CMA-ES

The elitist CMA-ES is a combination of the well-known $(1 + 1)$ selection scheme (Rechenberg 1973; Schwefel 1995; Beyer and Schwefel 2002) with the covariance matrix adaptation as proposed for the non-elitist CMA-ES by Hansen and Ostermeier (1996, 2001). In the elitist CMA-ES the adaptation of the covariance matrix employs the rank-one update from the original, non-elitist CMA-ES. However, the adaptation of the global step size is handled differently; the cumulative step size adaptation of the non-elitist CMA-ES is replaced by a success rule based step size control.

In this section we consider scalar fitness functions $f : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto f(\boldsymbol{x})$ to be minimized. We describe the original $(1 + 1)$-CMA-ES as proposed by Igel et al. (2006) and then incorporate the *Cholesky update* following Corollary 1. In the original algorithm, parent and offspring encode candidate solution vectors $\boldsymbol{x}_{\text{parent}}, \boldsymbol{x}_{\text{offspring}} \in \mathbb{R}^n$. We keep track of an averaged success rate $\overline{p}_{\text{succ}} \in [0, 1]$, the global step size $\sigma \in \mathbb{R}^+$, an evolution path $\boldsymbol{p}_c \in \mathbb{R}^n$, and the covariance matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$.

The algorithm is described in three routines. In the main part, termed $(1 + 1)$-CMA-ES and described in pseudo-code in the Algorithm 1 below, a new candidate solution is sampled and the parent solution $\boldsymbol{x}_{\text{parent}}$ is replaced depending on whether the new solution $\boldsymbol{x}_{\text{offspring}}$ is at least as good as $\boldsymbol{x}_{\text{parent}}$:

Algorithm 1: $(1 + 1)$-CMA-ES

initialize $\boldsymbol{x}_{\text{parent}}$, σ, $\boldsymbol{C}$, $\overline{p}_{\text{succ}} \leftarrow p_{\text{succ}}^{\text{target}}$, $\boldsymbol{p}_c \leftarrow \mathbf{0}$
repeat
  determine $\boldsymbol{A}$ such that $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}_{\text{offspring}} \leftarrow \boldsymbol{x}_{\text{parent}} + \sigma \boldsymbol{A}\boldsymbol{z}$
  `updateStepSize` $\left(\sigma, \mathbb{1}[f(\boldsymbol{x}_{\text{offspring}}) \leq f(\boldsymbol{x}_{\text{parent}})], \overline{p}_{\text{succ}}\right)$
  if $f\left(\boldsymbol{x}_{\text{offspring}}\right) \leq f\left(\boldsymbol{x}_{\text{parent}}\right)$ **then**
    $\boldsymbol{x}_{\text{parent}} \leftarrow \boldsymbol{x}_{\text{offspring}}$
    `updateCov` $\left(\boldsymbol{C}, \boldsymbol{A}\boldsymbol{z}, \overline{p}_{\text{succ}}, \boldsymbol{p}_c\right)$
until stopping criterion is met

Here the indicator function $\mathbb{1}[\cdot]$ is one if its argument is true and zero otherwise. Thus, $\mathbb{1}[f(\boldsymbol{x}_{\text{offspring}}) \leq f(\boldsymbol{x}_{\text{parent}})]$ is one if the last mutation has been successful and zero otherwise. The step size is updated depending on the success with a learning rate c_p $(0 < c_p \leq 1)$ using a target success rate $p_{\text{succ}}^{\text{target}}$:

Procedure `updateStepSize`$(\sigma, \lambda_{\text{succ}}, \overline{p}_{\text{succ}})$

1 $\overline{p}_{\text{succ}} \leftarrow (1 - c_p)\,\overline{p}_{\text{succ}} + c_p \lambda_{\text{succ}}$

2 $\sigma \leftarrow \sigma \cdot \exp\left(\frac{1}{d}\frac{\overline{p}_{\text{succ}} - p_{\text{succ}}^{\text{target}}}{1 - p_{\text{succ}}^{\text{target}}}\right)$

This procedure is rooted in the 1/5-success-rule proposed by Rechenberg (1973). The implementation follows a version proposed by Kern et al. (2004), smoothed by means of line 1 and generalized in that it can also be applied for $\lambda > 1$. It implements the well-known heuristic that the step size should be increased if the success rate (i.e., the fraction of offspring not worse than the parent) is high, and the step size should be decreased if the success rate is low. The damping parameter d controls the extend of the step size changes.

If the new candidate solution is not worse than its parent, the covariance matrix is updated as in the original CMA-ES (Hansen and Ostermeier 2001):

Procedure `updateCov`$(\boldsymbol{C}, \boldsymbol{y}, \overline{p}_{\text{succ}}, \boldsymbol{p}_c)$

1 **if** $\overline{p}_{\text{succ}} < p_{\text{thresh}}$ **then**

2 $\quad \boldsymbol{p}_c \leftarrow (1 - c_c)\,\boldsymbol{p}_c + \sqrt{c_c(2 - c_c)}\;\boldsymbol{y}$

3 $\quad \boldsymbol{C} \leftarrow (1 - c_{\text{cov}})\boldsymbol{C} + c_{\text{cov}} \cdot \boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}}$

4 **else**

5 $\quad \boldsymbol{p}_c \leftarrow (1 - c_c)\,\boldsymbol{p}_c$

6 $\quad \boldsymbol{C} \leftarrow (1 - c_{\text{cov}})\boldsymbol{C} + c_{\text{cov}} \cdot \left(\boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}} + c_c(2 - c_c)\boldsymbol{C}\right)$

The update of the evolution path $\boldsymbol{p}_c$ depends on the value of $\overline{p}_{\text{succ}}$. If the smoothed success rate $\overline{p}_{\text{succ}}$ is high, that is, above $p_{\text{thresh}} < 0.5$, the update of the evolution path $\boldsymbol{p}_c$ is stalled. This prevents a too fast increase of axes of $\boldsymbol{C}$ when the step size is far too small, for example, in a (close to) linear surrounding. If the smoothed success rate $\overline{p}_{\text{succ}}$ is low, the update of $\boldsymbol{p}_c$ is accomplished obeying an exponential smoothing. The constants c_c and c_{cov} ($0 \le c_{\text{cov}} < c_c \le 1$) are learning rates for the evolution path and the covariance matrix, respectively. The factor $\sqrt{c_c(2 - c_c)}$ normalizes the variance of $\boldsymbol{p}_c$ viewed as a random variable. The evolution path $\boldsymbol{p}_c$ is used to update the covariance matrix. The new covariance matrix is a weighted mean of the old covariance matrix and the outer product $\boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}}$. In the second case (line 5), the second summand in the update of $\boldsymbol{p}_c$ is missing and the length of $\boldsymbol{p}_c$ shrinks. Although of minor relevance, the term $c_c(2 - c_c)\boldsymbol{C}$ (line 6) compensates for this shrinking in $\boldsymbol{C}$.

Initial values are set to $\overline{p}_{\text{succ}} = p_{\text{succ}}^{\text{target}}$, $\boldsymbol{p}_c = \boldsymbol{0}$, and $\boldsymbol{C} = \boldsymbol{I}$, where $p_{\text{succ}}^{\text{target}}$ is given in Table 1. The initial candidate solution $\boldsymbol{x}_{\text{parent}} \in \mathbb{R}^n$ and the initial $\sigma \in \mathbb{R}^+$ must be chosen problem dependent. The optimum should presumably be within the cube $\boldsymbol{x}_{\text{parent}} \pm 2\sigma(1, \ldots, 1)^{\mathrm{T}}$. The

Table 1 Default parameters for the $(1 + 1)$-CMA evolution strategy

Step size control:

$$d = 1 + \frac{n}{2}, \quad p_{\text{succ}}^{\text{target}} = \frac{2}{11}, \quad c_p = \frac{1}{12}$$

Covariance matrix adaptation:

$$c_c = \frac{2}{n+2}, \quad c_{\text{cov}} = \frac{2}{n^2+6}, \quad p_{\text{thresh}} = 0.44$$

(external) strategy parameters of the $(1+1)$-CMA-ES and their default values, taken from Igel et al. (2007b) with λ set to one, are given in Table 1. Most default values are adopted from the precursor algorithms. In particular, the parameters for the covariance matrix adaptation are similar to those for the standard non-elitist CMA-ES. They have been validated by simulations on simple test functions for different dimensions, where the parameters were varied in a large interval. It has to be stressed that the parameters in Table 1 have to be rather viewed as constants and that in practice the user just needs to select the initial search point and the initial global step size (this is also true for the corresponding parameters of the ESs described below). We now discuss the setting for each parameter from Table 1 in turn.

The parameter $d \approx n/2$ is the damping parameter for the step size controlling the amplitude of step size changes. In order to avoid large stochastic fluctuations the damping parameter should be as large as possible. On the other hand it must be small enough to permit fast convergence. According to the `updateStepSize` procedure, the step size changing multiplier can range, for the given target success rate, between $\exp(-1/(4.5\,d)) \approx 0.80^{1/d}$ and $\exp(1/d) \approx 2.7^{1/d}$. The step-size changing factor is monotonous in the success rate. Given $p_{\text{succ}}^{\text{target}} = 2/11$ and $d = n/2$, we get for success rates of 0, 1/10, 1/7, and 1/5.5, the respective changing factors of about 0.64, 0.82, 0.91, and 1 (constant step size) in n iteration steps. The factors justify the choice for d. A step size reduction of approximately 0.82 in n iterations corresponds to the maximal possible convergence rate that can be achieved with an $(1+1)$-ES and optimal step size (Rechenberg 1973).

The parameter $p_{\text{succ}}^{\text{target}} \approx 0.2$ is the target success rate. A target success rate of 1/5 is well-established for the $(1+1)$-ES (Beyer and Schwefel 2002) and was derived from optimal success rates on simple fitness functions (Schumer and Steiglitz 1968; Rechenberg 1973). We use a slightly smaller target success rate, because it generally leads to a larger adapted step size, which in turn is more likely to escape local optima, though we believe that this effect is small.

The parameter $c_p \approx 0.1$ is the success rate averaging parameter, that is, the weight for the most recent value in the exponential smoothing of success events. The weight is chosen such that the success rate is smoothed over a significant period of about fifteen iterations, that is, a period with about two to three expected successes. While a large value for c_p leads to larger stochastic fluctuations in σ, a small value impedes the response time of the adaptation process.

The parameter $c_{\text{c}} \approx 2/n$ is the weight for the most recent entry in the exponential smoothing for the evolution path that is used in the covariance matrix update (it appears in the decay term as prefactor $1 - c_{\text{c}}$). The backward time horizon of the cumulation process is approximately c_{c}^{-1}. Only for $c_{\text{c}}^{-1} \propto n$ the performance scale-up of the optimization on ridge like functions becomes linear with n (Hansen and Ostermeier 2001) which rules out considerably larger values, for example $c_{\text{c}} = \frac{1}{\sqrt{n}}$. Too small values for c_{c} on the other hand would require an undesirable reduction of the learning rate for the covariance matrix.

The parameter $c_{\text{cov}} \approx 2/(n^2+6)$ is the learning rate for the covariance matrix, where c_{cov}^{-1} reflects the degrees of freedom of the covariance matrix with an additional adjustment for small n.

The parameter $p_{\text{thresh}} < 0.5$ is the threshold for stalling the update of evolution path and covariance matrix. Its value is chosen close to, but significantly smaller than one half: the update is stalled for large success probabilities. A success probability of one half occurs in a linear environment and will rarely be exceeded.

Now we apply Corollary 1 and replace the covariance matrix update by the Cholesky update. The new main routine reads:

Algorithm 2: $(1+1)$-Cholesky-CMA-ES

initialize $\boldsymbol{x}_{\text{parent}}$, σ, $\boldsymbol{A}$, $\boldsymbol{A}_{\text{inv}}$, $\overline{p}_{\text{succ}} \leftarrow p_{\text{succ}}^{\text{target}}$, $\boldsymbol{p}_c \leftarrow \boldsymbol{0}$
repeat
  $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{x}_{\text{offspring}} \leftarrow \boldsymbol{x}_{\text{parent}} + \sigma \boldsymbol{A}\boldsymbol{z}$
  `updateStepSize` $\left(\sigma, \mathbb{1}[f(\boldsymbol{x}_{\text{offspring}}) \leq f(\boldsymbol{x}_{\text{parent}})], \overline{p}_{\text{succ}}\right)$
  if $f\left(\boldsymbol{x}_{\text{offspring}}\right) \leq f\left(\boldsymbol{x}_{\text{parent}}\right)$ **then**
    $\boldsymbol{x}_{\text{parent}} \leftarrow \boldsymbol{x}_{\text{offspring}}$
    `updateCholesky` $\left(\boldsymbol{A}, \boldsymbol{A}_{\text{inv}}, \boldsymbol{A}\boldsymbol{z}, \overline{p}_{\text{succ}}, \boldsymbol{p}_c\right)$
until stopping criterion is met

In the $(1+1)$-Cholesky-CMA-ES the covariance matrix is never explicitly calculated and the update of the covariance is replaced by the corresponding update of the Cholesky factors and their inverse:

Procedure `updateCholesky` $(\boldsymbol{A}, \boldsymbol{A}_{\text{inv}}, \boldsymbol{z}, \overline{p}_{\text{succ}}, \boldsymbol{p}_c)$

if $\overline{p}_{\text{succ}} < p_{\text{thresh}}$ **then**
  $\boldsymbol{p}_c \leftarrow (1-c_c)\boldsymbol{p}_c + \sqrt{c_c(2-c_c)}\,\boldsymbol{z}$
  $\alpha \leftarrow (1-c_{\text{cov}})$
else
  $\boldsymbol{p}_c \leftarrow (1-c_c)\boldsymbol{p}_c$
  $\alpha \leftarrow (1-c_{\text{cov}}) + c_{\text{cov}} \cdot c_c(2-c_c)$
$\beta \leftarrow c_{\text{cov}}$
$\boldsymbol{w} \leftarrow \boldsymbol{A}_{\text{inv}} \cdot \boldsymbol{p}_c$
$\boldsymbol{A} \leftarrow \sqrt{\alpha}\boldsymbol{A} + \frac{\sqrt{\alpha}}{\|\boldsymbol{w}\|^2}\left(\sqrt{1+\frac{\beta}{\alpha}\|\boldsymbol{w}\|^2} - 1\right)\boldsymbol{p}_c\boldsymbol{w}^{\mathrm{T}}$
$\boldsymbol{A}_{\text{inv}} \leftarrow \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_{\text{inv}} - \frac{1}{\sqrt{\alpha}\|\boldsymbol{w}\|^2}\left(1 - \frac{1}{\sqrt{1+\frac{\beta}{\alpha}\|\boldsymbol{w}\|^2}}\right)\boldsymbol{w}\left[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_{\text{inv}}\right]$

The resulting algorithm is equivalent to the original version, except that the computational complexity of a single generation is reduced from $\Theta(n^3)$ to $\Theta(n^2)$. The memory requirements of both variants of the elitist CMA-ES are the same. A further advantage of the $(1+1)$-CMA-ES with Cholesky update, termed $(1+1)$-Cholesky-CMA-ES in the remainder of this article, is its simple implementation.

All considerations in this section are not restricted to the case of a single offspring, but also apply to the general $(1+\lambda)$-CMA-ES sampling λ offspring in each generation. The $(1+\lambda)$-CMA-ES is described by Igel et al. (2007a), and it is straightforward to derive the corresponding $(1+\lambda)$-Cholesky-CMA-ES.

4.1.1 Experimental evaluation

In this section, we present empirical results showing the validity and efficiency of our new update scheme. All experiments have been implemented using the Shark machine learning library (Igel et al. 2008).

Table 2 Single-objective benchmark functions used in our experiments. We define $\boldsymbol{y} = \boldsymbol{O}\boldsymbol{x}$, where $\boldsymbol{O} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix created randomly anew for each trial (each column vector is uniformly distributed on the unit hypersphere). In the noisy fitness function $f_{\text{sphereCauchy}}$, ξ denotes a random variable drawn according to the standard Cauchy distribution

Noisy Sphere, $n = 20$, $I_{\text{init}} = [0.1, 0.3]^n$
$f_{\text{sphereCauchy}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 + \frac{1}{2n}\xi \sum_{i=1}^{n} x_i^2$
Ellipsoid, $n = 20$, $I_{\text{init}} = [0.1, 0.3]^n$, $c = 10^6$
$f_{\text{elli}}(\boldsymbol{x}) = \sum_{i=1}^{n} c^{\frac{i-1}{n-1}} y_i^2$
Cigar, $n = 20$, $I_{\text{init}} = [0.1, 0.3]^n$, $c = 10^6$
$f_{\text{cigar}}(\boldsymbol{x}) = y_1^2 + \sum_{i=2}^{n} c y_i^2$
Generalized Rosenbrock's function, $n = 20$, $I_{\text{init}} = [0.1, 0.3]^n$
$f_{\text{Rosenbrock}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2)$

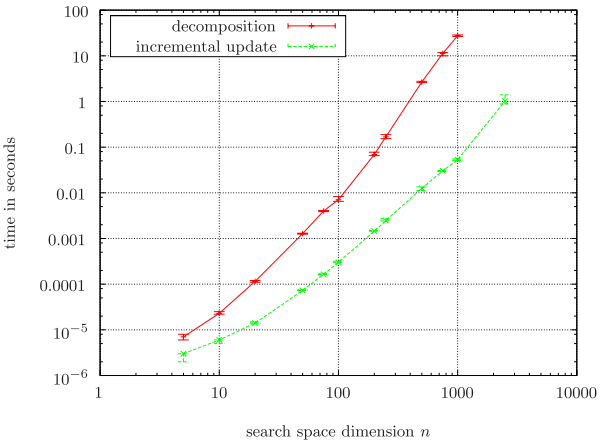

Fig. 2 Comparison of the time needed for covariance updates in the elitist CMA-ESs depending on the problem dimension n. We measured the average time of the covariance matrix updates on f_{elli}, based on the first 100 covariance updates. Shown is the median, the error bars indicate the 10th and 90th percentile. The standard elitist CMA-ES uses an efficient Cholesky decomposition for each update, while our new method performs the updates of the Cholesky factor and its inverse incrementally. Note the double log scale

Computational complexity We experimentally demonstrate the gain in computational efficiency achieved by using the new incremental update rule compared to the original $(1 + 1)$-CMA-ES, which decomposes the covariance matrix in every iteration. We measured the average time (on an Intel® Xeon™ CPU with 2.40 GHz running Linux) needed for a covariance matrix update on the objective function f_{elli} (see Table 2) depending on the problem dimension n. The average was computed over the first 100 updates.

The original elitist CMA-ES used a Cholesky factorization algorithm (Grewal and Andrews 1993, Sect. 6.4, p. 217), which was efficiently implemented.

The results are presented in Fig. 2. The double log scale of the plot nicely shows the different exponents of the polynomial scaling of the algorithms. Already for 20 dimensions the new update is almost ten times faster. For $n = 1000$ the factor is about five hundred.

Stability We performed several experiments on different objective functions to check the numerical stability of our update rules. Because the inverse of the Cholesky factor is updated independently of the Cholesky factor, one must suspect that they drift apart. Therefore, we monitored the evolution of $\langle \boldsymbol{A}\boldsymbol{A}_{\text{inv}} - \boldsymbol{I} \rangle_{\text{F}}$ during the optimization. Here $\langle \cdot \rangle_{\text{F}}$ denotes the Frobenius matrix norm. As a baseline comparison we additionally monitored $\langle \boldsymbol{A}\boldsymbol{A}^{-1} - \boldsymbol{I} \rangle_{\text{F}}$, where the inverse $\boldsymbol{A}^{-1}$ of $\boldsymbol{A}$ is computed in every step using an accurate standard method (based on singular value decomposition, SVD).

We considered the quadratic test problem f_{elli} having a Hessian matrix with condition number of 10^6 for $n = 3$ and $n = 200$, and the Rosenbrock function $f_{\text{Rosenbrock}}$ that requires

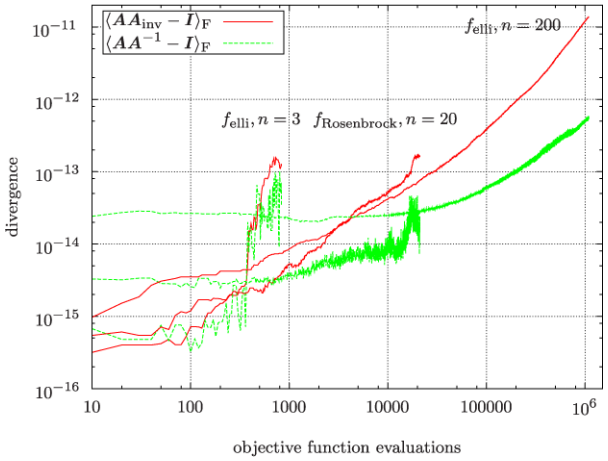

Fig. 3 Accuracy of the incremental covariance matrix update compared to standard matrix inversion. Shown are typical trials on f_{elli} with $n = 3$ and $n = 200$ as well as on $f_{\text{Rosenbrock}}$ with $n = 20$. The trials were stopped after an objective function value smaller than 10^{-15} was reached. The divergence was checked every 10th generation. Note the double log scale

adaptation to a changing local metric for $n = 20$. As done in practice, we started with the unit covariance matrix, that is, perfect initial decomposition.

The results are shown in Fig. 3. Both methods lose accuracy with time. Indeed, in the first $20n$ generations the iterative updates are even more accurate than the inversion algorithm. After that, the iterative updates accumulate more errors, but even after $20n^2$ generations the error never exceeds 10^{-11}.

Thus, we regard our update rules as highly accurate and numerically stable. Of course, it is possible to reset the iterative update every $\Omega(n)$ iterations computing the matrix inversion explicitly without affecting the run-time complexity. Our results show that this is not necessary at least before $50n^2$ iterations.

Necessity of the evolution path The new $(1 + 1)$-Cholesky-CMA-ES is a significant improvement over the method previously proposed by the authors in Igel et al. (2006). Both methods implement an incremental, computational efficient update of the Cholesky matrix, but only the new one can utilize an evolution path. Igel et al. (2006) give several examples of the decline in optimization performance when the evolution path is omitted. Here, we replicated one of these experiments comparing the $(1 + 1)$-Cholesky-CMA-ES to the previously presented work.

Figure 4 shows the fitness evolution on the 20-dimensional f_{elli} benchmark function based on 25 independent trials. The new method can learn the topology of the objective function about four times faster (within about 13 000 function evaluations). Also the second phase, the log-linear convergence to the optimum, is slightly faster. The shown result is, to our experience, representative for different dimensionalities and also for many objective functions.

The evolution path facilitates especially the learning of a single elongated axis in the shape of the distribution, the main task on the cigar benchmark function. Figure 5 shows the number of function evaluations to reach the target function value 10^{-15} on f_{cigar} depending on dimension n. With evolution path, the graph resembles virtually $300n$, the scaling is linear. Without evolution path the graph is close to $150n^{1.8}$, the scaling is almost quadratic.

On some objective functions only small differences can be observed and omitting the evolution path can even be (slightly) advantageous (Igel et al. 2006). Nevertheless in general, by exploiting the information on correlations between the steps in the evolution path, the new $(1 + 1)$-Cholesky-CMA-ES considerably outperforms the old one.

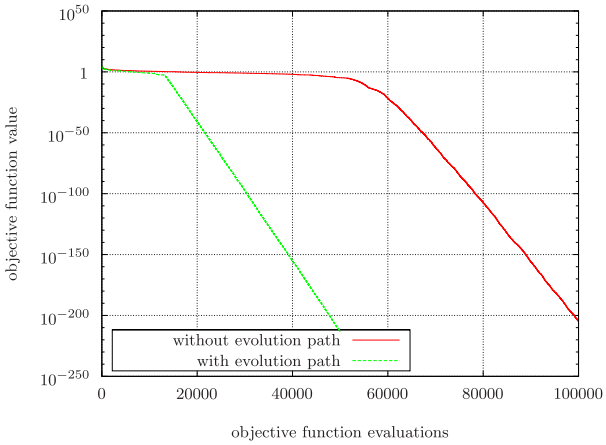

Fig. 4 Comparison of the elitist $(1+1)$-CMA-ES on the 20-dimensional f_{elli} with and without evolution path. The curves show the medians of 25 trials. Both methods use an efficient incremental rank-one update, but the new algorithm learns the underlying metric about four times faster

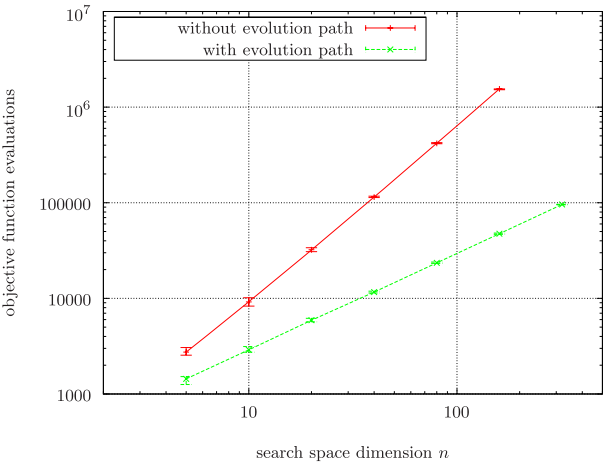

Fig. 5 Comparison of the elitist $(1+1)$-CMA-ES on f_{cigar} with and without evolution path. We measured the number of iterations needed to reach an objective function value smaller than 10^{-15}. The curves show the medians of 25 trials for $n < 100$ and of 5 trials for $n > 100$ on a double log scale. Error bars denote the 10th and 90th percentile

4.2 Multi-objective CMA-ES

Multi-objective optimization (MOO, or vector optimization) is concerned with the simultaneous optimization of multiple (scalar) objectives (e.g., Miettinen 1999; Deb 2001; Jin 2006). The goal of MOO is usually to find or to approximate the set of Pareto-optimal solutions. A solution is Pareto-optimal if it cannot be improved in one objective without getting worse in another one. A diverse set of Pareto-optimal solutions provides insights into the trade-offs between the objectives and serves as the basis for (human) decision-making. In recent years, evolutionary multi-objective algorithms have become popular for MOO (e.g., Deb 2001; Jin 2006).

We consider real-valued MOO with m objectives. Each point $\boldsymbol{x}$ of the search space is assigned m objective function values $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m, \boldsymbol{x} \mapsto (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^{\mathrm{T}}$. We say that a solution $\boldsymbol{x}$ dominates a solution $\boldsymbol{x}'$ and write $\boldsymbol{x} \prec \boldsymbol{x}'$, if and only if $\boldsymbol{x}$ is at least as good as $\boldsymbol{x}'$ in each objective and is strictly better than $\boldsymbol{x}'$ in at least one objective.

The multi-objective CMA-ES (MO-CMA-ES) considers a population of μ solutions, where every solution reproduces and updates its search strategy as in the $(1+1)$-CMA-ES. The main additional problem to solve is the question of how to rank a set of solutions in order to determine the best μ candidate solutions forming the next parent population. The

general approach to multi-objective ranking as taken in the MO-CMA-ES is introduced next, before we describe the search algorithm using the new update rule.

4.2.1 Ranking vector-valued solutions

The ranking in the MO-CMA-ES relies on non-dominated sorting as in the NSGA-II (Deb 2001; Deb et al. 2002) and on the order induced by the contributing hypervolume as in the SMS-EMOA (Beume et al. 2007).

In a first step, the elements in a population X of candidate solutions are ranked according to their level of non-dominance. We denote the non-dominated solutions in X by $\mathrm{ndom}(X) = \{\boldsymbol{x} \in X \mid \nexists \boldsymbol{x}' \in X : \boldsymbol{x}' \prec \boldsymbol{x}\}$, where $\boldsymbol{x}' \prec \boldsymbol{x}$ means that $\boldsymbol{x}'$ dominates $\boldsymbol{x}$. The Pareto front of X is then given by $\{(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathrm{ndom}(X)\}$. The elements in $\mathrm{ndom}(X)$ get rank 1. The other ranks are defined recursively by considering the set without the solutions with lower ranks (cf. Deb et al. 2002; Igel et al. 2007a). Let $\mathrm{dom}_0(X) = X$, $\mathrm{dom}_l(X) = \mathrm{dom}_{l-1}(X) \setminus \mathrm{ndom}_l(X)$, and $\mathrm{ndom}_l(X) = \mathrm{ndom}(\mathrm{dom}_{l-1}(X))$ for $l \in \{1, \ldots\}$. For $\boldsymbol{x} \in X$ the level of non-dominance $\mathrm{r}(\boldsymbol{x}, X)$ is i iff $\boldsymbol{x} \in \mathrm{ndom}_i(X)$. Level of non-dominance is the first sorting criterion.

A second sorting criterion is needed to rank the solutions that have the same level of non-dominance. This criterion is very important, as usually (in particular in the optimization of continuous objective functions) after some generations the number of non-dominated solutions in the population exceeds the number of solutions to be selected.

In the MO-CMA-ES, the contributing hypervolume serves as second sorting criterion. The hypervolume measure or $\mathcal{S}$-metric was introduced by Zitzler and Thiele (1998) in the domain of evolutionary MOO. For a reference point $\boldsymbol{x}_{\mathrm{ref}}$, the hypervolume $\mathcal{S}_{\boldsymbol{x}_{\mathrm{ref}}}(X')$ of a set X' is defined as volume of the union of hypercuboids

$$\mathcal{S}_{\boldsymbol{x}_{\mathrm{ref}}}(X') = \bigcup_{\boldsymbol{x} \in \mathrm{ndom}(X')} \{(f_1(\boldsymbol{x}'), \ldots, f_m(\boldsymbol{x}')) \mid \boldsymbol{x} \prec \boldsymbol{x}' \prec \boldsymbol{x}_{\mathrm{ref}}\}.$$

The contributing hypervolume $\Delta_{\mathcal{S}}(\boldsymbol{x}, X')$ of a point $\boldsymbol{x} \in \mathrm{ndom}(X')$ is given accordingly by $\Delta_{\mathcal{S}}(\boldsymbol{x}, X') := \mathcal{S}_{\boldsymbol{x}_{\mathrm{ref}}}(X') - \mathcal{S}_{\boldsymbol{x}_{\mathrm{ref}}}(X' \setminus \{\boldsymbol{x}\})$. The rank $s(\boldsymbol{x}, X')$ of an individual a can be defined recursively based on its contribution to the hypervolume, where ties are broken at random. The individual contributing least to the hypervolume of X' gets the lowest rank. The individual contributing least to the hypervolume of X' without the individual with the lowest rank is assigned the second lowest rank and so on. We call $\boldsymbol{x} \in X'$ a boundary element if $\Delta_{\mathcal{S}}(\boldsymbol{x}, X')$ depends on the choice of the reference point $\boldsymbol{x}_{\mathrm{ref}}$. The point $\boldsymbol{x}_{\mathrm{ref}}$ is chosen such that all elements in X' dominate $\boldsymbol{x}_{\mathrm{ref}}$ and that $\Delta_{\mathcal{S}}(\boldsymbol{x}, X') > \Delta_{\mathcal{S}}(\boldsymbol{x}', X')$ holds for any boundary element $\boldsymbol{x} \in X'$ and any non boundary element $\boldsymbol{x}' \in X'$. That is, the individuals at the "boundaries" of the Pareto front of X' are preferably selected. Let a lower rank be worse. Formally (assuming that argmin breaks ties randomly), for $\boldsymbol{x} \in \mathrm{ndom}(X')$ it holds $s(\boldsymbol{x}, X') = 1$ if $\boldsymbol{x} = \mathrm{argmin}_{\boldsymbol{x}' \in X'}\{\Delta_{\mathcal{S}}(\boldsymbol{x}', X')\}$ and $s(\boldsymbol{x}, X') = k$ if $\boldsymbol{x} = \mathrm{argmin}_{\boldsymbol{x}' \in X'}\{\Delta_{\mathcal{S}}(\boldsymbol{x}', X' \setminus \{\boldsymbol{x}'' \mid s(\boldsymbol{x}'', X') < k\})\}$. Based on this ranking and the level of non-dominance the relation

$$\begin{aligned}\boldsymbol{x} \prec_X \boldsymbol{x}' \Leftrightarrow\ & \mathrm{r}(\boldsymbol{x}, X) < \mathrm{r}(\boldsymbol{x}', X) \text{ or} \\ & \big[(\mathrm{r}(\boldsymbol{x}, X) = \mathrm{r}(\boldsymbol{x}', X)) \wedge (s(\boldsymbol{x}, \mathrm{ndom}_{\mathrm{r}(\boldsymbol{x}, X)}(X)) > s(\boldsymbol{x}', \mathrm{ndom}_{\mathrm{r}(\boldsymbol{x}', X)}(X)))\big],\end{aligned}$$

is defined on X. That is, $\boldsymbol{x}$ is better than $\boldsymbol{x}'$ when compared using $\prec_X$ if and only if either $\boldsymbol{x}$ has a better level of non-dominance or $\boldsymbol{x}$ and $\boldsymbol{x}'$ are on the same level but $\boldsymbol{x}$ contributes more to the hypervolume when considering the points at that level of non-dominance.

Computing the hypervolume for many objectives is computationally demanding (Klee 1977; Overmars and Yap 1991). However, for $1 < m \leq 3$ special algorithms exist which solve this problem in $O(\mu^{m-2} \ln \mu)$, where μ is the number of points in the objective space (Fonseca et al. 2006). For $m > 3$, the current best algorithm scales with $O(\mu \ln \mu + \mu^{m/2})$ (Beume and Rudolph 2006). If computing the contributing hypervolume for selection turns out to be too time consuming in an application, the contributing hypervolume can be replaced by the change in quality measured by the ϵ-indicator (Zitzler et al. 2003), which can be computed efficiently.

4.2.2 The MO-CMA-ES

In the following, we describe the MO-CMA-ES working on Cholesky decompositions of the covariance matrices, for the original version we refer to Igel et al. (2007a, 2007b). The variant described here uses steady-state selection, that is, only one offspring is generated per generation. We consider steady-state selection where all μ members of the population are potential parents (i.e., (μ+1)-selection). This corresponds to the selection scheme used by Emmerich et al. (2005) and Beume et al. (2007). Conceptually, this selection scheme has the advantage that the problem of selecting the μ out of $\mu + \lambda$ possible points such that the hypervolume is maximized can be solved easily for $\lambda = 1$.

An individual in the MO-CMA-ES is a 6-tuple

$$a = \left[\boldsymbol{x}, \overline{p}_{\text{succ}}, \sigma, \boldsymbol{p}_c, \boldsymbol{A}, \boldsymbol{A}_{\text{inv}}\right],$$

where, $\boldsymbol{x} \in \mathbb{R}^n$ is the point in the search space, $\overline{p}_{\text{succ}} \in \mathbb{R}_0^+$ the average success rate, $\sigma \in \mathbb{R}^+$ the global step size, $\boldsymbol{p}_c \in \mathbb{R}^n$ the evolution path, and $\boldsymbol{A}$ and $\boldsymbol{A}_{\text{inv}}$ are a Cholesky factor of the covariance matrix and its inverse. The steady-state MO-CMA algorithm can be described as follows.

Algorithm 3: steady-state MO-CMA

1 initialize a_k for $k = 1, \ldots, \mu$
2 $Q \leftarrow \left\{a_k \,\middle|\, 1 \leq k \leq \mu\right\}$
3 **repeat**
4 $\quad i \leftarrow \mathcal{U}(1, \mu)$
5 $\quad a \leftarrow Q_{\prec:i}$
6 $\quad a' \leftarrow a$
7 $\quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
8 $\quad \boldsymbol{x}' \leftarrow \boldsymbol{x} + \sigma \boldsymbol{A} \boldsymbol{z}$
9 $\quad Q \leftarrow Q \cup \{a'\}$
10 $\quad$ **if** $a' \neq Q_{\prec:(\mu+1)}$ **then**
11 $\quad\quad$ `updateStepSize` $\left(\sigma', \mathbb{1}\left[a' \prec_Q a\right], \overline{p}'_{\text{succ}}\right)$
12 $\quad\quad$ `updateCholesky` $\left(\boldsymbol{A}', \boldsymbol{A}'_{\text{inv}}, \dfrac{\boldsymbol{x}' - \boldsymbol{x}}{\sigma}, \overline{p}'_{\text{succ}}, \boldsymbol{p}'_c\right)$
13 $\quad$ `updateStepSize` $\left(\sigma, \mathbb{1}\left[a' \prec_Q a\right], \overline{p}_{\text{succ}}\right)$
14 $\quad Q \leftarrow Q \setminus Q_{\prec:(\mu+1)}$
15 **until** stopping criterion is met

At the beginning the population Q is initialized with μ parents. In the iteration loop, first a parent is chosen randomly in lines 4–5, where $\mathcal{U}(1, \mu)$ denotes the discrete uniform distrib-

ution over $\{1, \ldots, \mu\}$ and $Q_{\prec:i}$ is the ith best individual in Q according to $\prec_Q$. One offspring $a' = [\boldsymbol{x}', \overline{p}'_{\text{succ}}, \sigma', \boldsymbol{p}'_c, \boldsymbol{A}', \boldsymbol{A}'_{\text{inv}}]$ is generated from the parent and added to the current population Q (lines 6–9). Then, step size and Cholesky factor of the offspring are updated, in case it is not the worst individual (lines 10–12). Finally, the step size of the parent is updated and the worst individual is removed from the population Q.

The described algorithm is equivalent to the MO-CMA-ES from Igel et al. (2007a) while reducing its computational complexity from $\Theta(n^3)$ to $\Theta(n^2)$ per generation step. Igel et al. (2007b) showed that using the CMA-ES covariance update rule without evolution path decreases the performance of the algorithm. However, in the new method an incremental covariance update is realized in combination with an evolution path. Hence, the search behaviors of the original generational and steady-state MO-CMA-ESs do not change when using the incremental update rule. All performance comparisons conducted for the original algorithms carry over to the new variants with computationally efficient covariance matrix update. This was validated empirically on the benchmark scenario described by Igel et al. (2007b).

The $(\mu + 1)$-selection has proven to provide excellent performance on a great number of benchmark problems (Igel et al. 2007a, 2007b) compared to NSGA-II (Deb et al. 2002) and NSDE (Iorio and Li 2005). Recently, Voß et al. (2008) compared the MO-CMA-ES with scalarization approaches for multi-objective optimization. For scalarization, the Tchebycheff method as well as the weighted-sum approach were considered (Miettinen 1999) and the $(1 + 1)$-CMA-ES served as single objective optimizer. The MO-CMA-ES clearly outperformed the scalarization approaches. For the detailed performance evaluation of the MO-CMA-ESs we refer to the comparisons by Igel et al. (2007a, 2007b) and Voß et al. (2008).

4.3 Non-elitist CMA-ES

In this section, we combine the incremental Cholesky update with a slightly simplified non-elitist $(\mu/\mu_{\text{w}}, \lambda)$-CMA-ES (Hansen and Ostermeier 2001; Beyer 2007; Hansen et al. 2008), where μ/μ_{w} denotes weighted recombination of μ parental individuals. Non-elitism avoids systematic overvaluation[6] in the presence of noise, and is thus of particular importance for many machine learning applications, as argued in the introduction.

The resulting new algorithm is a computationally more efficient variant of the original CMA-ES with rank-one updating (Hansen and Ostermeier 2001). The need to simplify the original algorithm is due to its step size adaptation and is discussed in the following.

4.3.1 Cumulative step size adaptation

In the $(\mu/\mu_{\text{w}}, \lambda)$-CMA-ES the global step size σ is adjusted using *cumulative step size adaptation* (CSA, sometimes denoted as *path length control*). Let the offspring be generated using weighted global intermediate recombination of the selected parents, followed by Gaussian mutation with covariance matrix $\sigma^2\boldsymbol{C} = \sigma^2\boldsymbol{A}\boldsymbol{A}^{\text{T}}$. That is, prior to the mutation, the weighted center of mass $\langle \boldsymbol{x} \rangle_{\boldsymbol{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$ is computed, where $\boldsymbol{x}_{i:\lambda}$ is the ith best *parent* and $\boldsymbol{w} = (w_1, \ldots, w_\mu)^{\text{T}} \in \mathbb{R}^\mu$ are weighting coefficients with $w_1 \geq w_2 \geq \cdots \geq w_\mu$ and

[6]Overvaluation (Arnold 2002) refers to the effect that in the course of evolution individuals with their fitness sampled *from the tail* of the noise distribution are sustained. In order to compete with the previous parent, an individual needs to experience a similarly extreme noise event. This leads to continuously decreasing success rates and reduced progress.

$\|\boldsymbol{w}\|_1 = 1$. Accordingly, we define $\langle z \rangle_{\boldsymbol{w}} = \sum_{i=1}^{\mu} w_i z_{i:\lambda}$ as the weighted mean of the μ realizations of the standard normally distributed random vector that led to the μ best offspring. Thus, the step taken by the weighted mean of the population can be expressed as $\sigma \boldsymbol{A} \langle z \rangle_{\boldsymbol{w}}$. The matrix $\sigma \boldsymbol{A}$ scales and rotates the random vectors. We can remove all scaling effects by considering the orthogonal matrix $\boldsymbol{B}$ instead of $\sigma \boldsymbol{A}$, when the Cholesky factor $\boldsymbol{A}$ is given in the form $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{D}$ with $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ orthogonal and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ diagonal.

Standard CSA (Hansen and Ostermeier 2001; Hansen et al. 2008) keeps track of a "conjugate" evolution path

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\,\boldsymbol{B}\langle z \rangle_{\boldsymbol{w}} \tag{7}$$

with learning rate $c_\sigma \in]0, 1]$. The evolution path $\boldsymbol{p}_\sigma$ is a weighted sum of random vectors originally distributed according to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Under random selection (when $\boldsymbol{x}$ and $f(\boldsymbol{x})$ are independent) the normalization in (7) ensures that $\boldsymbol{p}_\sigma$, viewed as a random variable, is also distributed according to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (this is explained in detail in the review by Hansen 2006).

The factor $\sqrt{\mu_{\text{eff}}}$ compensates for the loss of variance due to computing the weighted center of mass during recombination. The effective parent number, μ_{eff}, also called *variance effective selection mass*, is given by $\mu_{\text{eff}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$. It is used for both step size control and covariance matrix adaptation.

In $\boldsymbol{p}_\sigma$, the influence of previous steps decays exponentially fast, controlled by the learning rate c_σ. The conjugate evolution path can (and should) be learned on a faster timescale than the covariance matrix. Hansen et al. (2008) propose the following learning rate

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 3}. \tag{8}$$

The update of the global step size σ is realized by

$$\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\hat{\chi}_n} - 1\right)\right). \tag{9}$$

Here $\hat{\chi}_n = E\,\|\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\|$ is the expected length of an n-dimensional, normally distributed random vector with covariance matrix $\boldsymbol{I}$. The damping parameter d_σ decouples the adaptation rate from the strength of the variation. Because of the proper normalization in (7), the expected length of $\boldsymbol{p}_\sigma$ would be $\hat{\chi}_n$ under random selection. Therefore, the global step size is increased if the steps leading to selected individuals are larger and/or more correlated than expected and decreased if they are smaller and/or more anticorrelated than expected in the absence of selection pressure.

Unfortunately, the incremental update rules of Corollary 1 do not provide the matrix $\boldsymbol{B}$ needed in (7).[7] Therefore, we omit $\boldsymbol{B}$ in the update of the global step size σ, similarly as in (Beyer 2007),

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\langle z \rangle_{\boldsymbol{w}}. \tag{10}$$

In this formulation, $\boldsymbol{p}_\sigma$ viewed as a random variable would be still distributed according to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with expected length $\hat{\chi}_n$ under random selection. Hence, large or small selected

[7] In general, the incrementally updated Cholesky factor $\boldsymbol{A}$ is not symmetric. A symmetric Cholesky factor with $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^{\mathrm{T}}$ would allow for an alternative formulation of (7) by replacing $\boldsymbol{B}\langle z \rangle_{\boldsymbol{w}}$ with $\sigma^{-1}\boldsymbol{A}_{\text{inv}}(\langle \boldsymbol{x} \rangle_{\boldsymbol{w}}^{(g+1)} - \langle \boldsymbol{x} \rangle_{\boldsymbol{w}}^{(g)})$, thus not requiring $\boldsymbol{B}$.

steps still lead to increase or decrease of the step size, respectively. However, the identification of the degree of correlation of successive steps becomes an approximation, because $\boldsymbol{B}$ changes over time. Compared to the decay of the conjugate evolution path $\boldsymbol{p}_\sigma$ in (10) the changes in $\boldsymbol{B}$ are small. To further diminish their effect, we increase the learning rate for the evolution path to

$$c_\sigma = \frac{\sqrt{\mu_{\text{eff}}}}{\sqrt{n} + \sqrt{\mu_{\text{eff}}}}. \tag{11}$$

In (11), the dependency of c_σ on n is taken to its upper limit $\frac{1}{\sqrt{n}}$ (Hansen 1998). Compared to (8), the interdependency between μ_{eff} and n is preserved, in that for $\mu_{\text{eff}} \gg n$ the coefficient gets close to one.

When the Cholesky factor $\boldsymbol{A}$ remains constant, as it can be approximately the case in the final phase of optimization, the rules (9) and (10) result in the same update behavior for σ. Otherwise, we expect the approximation to be sufficiently accurate and only insignificantly affect the performance. This is confirmed by our experiments in Sect. 4.3.3.

4.3.2 Non-elitist CMA-ES with incremental Cholesky update

The simplified non-elitist $(\mu/\mu_{\text{w}}, \lambda)$-CMA-ES with Cholesky update, referred to as $(\mu/\mu_{\text{w}}, \lambda)$-Cholesky-CMA-ES in the following, is completely described in the following algorithm.

Algorithm 4: $(\mu/\mu_{\text{w}}, \lambda)$-Cholesky-CMA-ES

1 initialize σ, $\boldsymbol{x}_k$ for $k = 1, \ldots, \lambda$, $\boldsymbol{A} \leftarrow \boldsymbol{I}$, $\boldsymbol{A}_{\text{inv}} \leftarrow \boldsymbol{I}$, $\boldsymbol{p}_\sigma \leftarrow \boldsymbol{0}$, $\boldsymbol{p}_c \leftarrow \boldsymbol{0}$
2 **repeat**
3 $\quad \langle \boldsymbol{x} \rangle_{\boldsymbol{w}} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$
4 $\quad$ **for** $k = 1, \ldots, \lambda$ **do**
5 $\quad\quad z_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
6 $\quad\quad \boldsymbol{x}_k \leftarrow \langle \boldsymbol{x} \rangle_{\boldsymbol{w}} + \sigma \boldsymbol{A} \boldsymbol{z}_k$
7 $\quad \langle \boldsymbol{z} \rangle_{\boldsymbol{w}} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}$
8 $\quad \boldsymbol{p}_c \leftarrow (1 - c_{\text{c}}) \boldsymbol{p}_c + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \boldsymbol{A} \langle \boldsymbol{z} \rangle_{\boldsymbol{w}}$
9 $\quad \boldsymbol{v} \leftarrow \boldsymbol{A}_{\text{inv}} \cdot \boldsymbol{p}_c$
10 $\quad \boldsymbol{A}_{\text{inv}} \leftarrow \frac{1}{\sqrt{1-c_{\text{cov}}}} \boldsymbol{A}_{\text{inv}} - \frac{1}{\sqrt{1-c_{\text{cov}}} \|\boldsymbol{v}\|^2} \left(1 - \frac{1}{\sqrt{1 + \frac{c_{\text{cov}}}{1-c_{\text{cov}}} \|\boldsymbol{v}\|^2}} \right) \boldsymbol{v} \left[\boldsymbol{v}^{\text{T}} \boldsymbol{A}_{\text{inv}} \right]$
11 $\quad \boldsymbol{A} \leftarrow \sqrt{1 - c_{\text{cov}}} \boldsymbol{A} + \frac{\sqrt{1-c_{\text{cov}}}}{\|\boldsymbol{v}\|^2} \left(\sqrt{1 + \frac{c_{\text{cov}}}{1-c_{\text{cov}}} \|\boldsymbol{v}\|^2} - 1 \right) \boldsymbol{p}_c \boldsymbol{v}^{\text{T}}$
12 $\quad \boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma) \boldsymbol{p}_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \langle \boldsymbol{z} \rangle_{\boldsymbol{w}}$
13 $\quad \sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma\|}{\hat{\chi}_n} - 1 \right) \right)$
14 **until** stopping criterion is met

In each iteration, λ offspring are generated by weighted global intermediate recombination (lines 4–7). Then, instead of the covariance matrix update in the original algorithm, the Cholesky factor and its inverse are updated according to Corollary 1 (lines 8–11). Finally, the global step size σ is adapted by the simplified CSA (lines 12 and 13).

The parameters for the $(\mu/\mu_{\text{w}}, \lambda)$-Cholesky-CMA-ES are given in Table 3. They are taken from Hansen et al. (2008) for the case $\mu_{\text{cov}} = 1$, using the increased learning rate for the conjugate evolution path (11) instead of (8). We describe the reasoning behind the

Table 3 Parameters for the $(\mu/\mu_{\mathrm{w}}, \lambda)$-Cholesky-CMA-ES. All parameters equal the parameters of the original variant, except for c_σ

Selection and recombination:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu = \lfloor \lambda/2 \rfloor, \quad w_i = \frac{\ln(\mu+1) - \ln i}{\mu \ln(\mu+1) - \sum_{j=1}^{\mu} \ln j} \quad \text{for } i = 1, \ldots, \mu$$

Step size control:

$$c_\sigma = \frac{\sqrt{\mu_{\mathrm{eff}}}}{\sqrt{n} + \sqrt{\mu_{\mathrm{eff}}}}, \quad d_\sigma = 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{\mathrm{eff}} - 1}{n+1}} - 1\right) + c_\sigma$$

Covariance matrix adaptation:

$$c_{\mathrm{c}} = \frac{4}{n+4}, \quad c_{\mathrm{cov}} = \frac{2}{(n+\sqrt{2})^2}$$

choice of selection and recombination parameters, and of d_σ in the following. The remaining parameters were already discussed above (parameter c_σ) or in Sect. 4.1 (parameters c_{c} and c_{cov}).

The parameter λ is the population size and its default value slowly increases with the dimension. The small setting for λ emphasizes fast convergence and reflects the point of view that an ES is a robust *local* search procedure. Smaller settings for λ can lead to a speed up, but too small settings endanger the working of the algorithm. With increasing population size the search becomes more and more global (given an adequate value for μ_{eff}). For considerably larger λ the rank-μ update for the covariance matrix should be used.

The parameters μ and $(w_i)_{i=1,\ldots,\mu}$ denote parent number and recombination weights. The chosen recombination weights approximate the optimal setting on the sphere function (Arnold 2006) for positive optimal weights. Negative weights are disregarded as, to our experience, they can be detrimental on non-spherical fitness functions. In general, in order to exploit the selection information effectively, the variance effective selection mass, μ_{eff}, should lie between $\lambda/5$ and $\lambda/2$. With the given setting, $\mu_{\mathrm{eff}} := 1/\sum_{i=1}^{\mu} w_i^2$ equals approximately $\lambda/3$.

The parameter $d_\sigma \approx 1$ controls the magnitude of step-size changes similar to d in Table 1. Equation (9) is formulated such that a good choice for d_σ does not (heavily) depend on c_σ. Smaller values for d_σ, for example $d_\sigma = 0.5$, can speed up the convergence. Too small values render the adaptation scheme unstable. Larger values will slow down the convergence without further harm. For large values of μ_{eff}, d_σ is increased and a rank-μ covariance matrix update should be used.

The original $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES requires an eigendecomposition of the covariance matrix. The eigendecomposition is more expensive and more difficult to implement than the Cholesky decomposition. As our main achievement, the $(\mu/\mu_{\mathrm{w}}, \lambda)$-Cholesky-CMA-ES as described in this section does not need any of these procedures.

4.3.3 Experimental evaluation

In order to evaluate the effect of the simplification of the CSA update, we compared the $(\mu/\mu_{\mathrm{w}}, \lambda)$-Cholesky-CMA-ES to the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES (using rank-one updating only) on 20-dimensional benchmark functions listed in Table 2.

The quadratic benchmark functions f_{elli} and f_{cigar} with rotated coordinate systems and condition numbers 10^6 are considered to investigate the learning of local metrics. They are particularly difficult, because they are non-separable and have high condition numbers. The

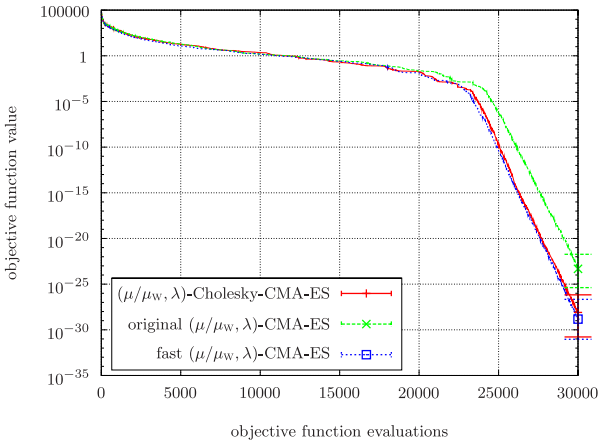

Fig. 6 Comparison of the $(\mu/\mu_W, \lambda)$-CMA-ES with rank-one updating and the simplified $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES on the benchmark function f_{elli} for $n = 20$. The $(\mu/\mu_W, \lambda)$-CMA-ES with original and increased learning rate c_σ are denoted by *original* and *fast* $(\mu/\mu_W, \lambda)$-CMA-ES, respectively. The trajectories show the medians of 25 trials, the final error bars the 10th and 90th percentile. The final difference between original $(\mu/\mu_W, \lambda)$-CMA-ES and $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES is significant with $p < 10^{-3}$ in the Wilcoxon rank-sum test. The curves of fast $(\mu/\mu_W, \lambda)$-CMA-ES and $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES can hardly be distinguished

ellipsoid f_{elli} is our prototypical benchmark function having eigenvalues distributed equidistantly on the log-scale. The function f_{cigar} tests the learning of a single elongated axis, see Sect. 4.1.1. The generalized Rosenbrock function $f_{\text{Rosenbrock}}$ with rotated coordinate system is difficult, because it is non-separable and the appropriate local metric changes in the course of optimization. Continuous adaptation of the covariance matrix is required. The sphere function with multiplicative Cauchy noise $f_{\text{sphereCauchy}}$ has been selected to demonstrate the performance in the case of noise. Replacing the Cauchy with a Gaussian noise distribution leads to similar results. As in all experiments, σ was initialized to $0.2/3$ (one third of the initialization interval I_{init} as given in Table 2). The $(\mu/\mu_W, \lambda)$-CMA-ES is run with the original value of c_σ for updating the "conjugate" evolution path and with the increased learning rate, (8) and (11), respectively.

The results, based on 25 trials per test function and algorithm, are presented in Figs. 6, 7, and 8. The algorithms performed very similar as expected. On f_{elli}, the $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES is slightly faster (about 5%, the difference is statistically significant) than the $(\mu/\mu_W, \lambda)$-CMA-ES with original learning rate c_σ. The difference can be entirely attributed to the new choice of parameter c_σ. The results on f_{cigar} are qualitatively the same as on f_{elli} and are therefore omitted. On the other two functions no relevant performance differences are observed. The shown results are representative and we have found, up to now, no case with a performance break down due to the simplification of CSA.

The standard $(\mu/\mu_W, \lambda)$-CMA-ES that includes the rank-μ update, reaches the function value of 10^{-10} on f_{elli} and $f_{\text{Rosenbrock}}$ around 20 % faster, and it performs on par on $f_{\text{sphereCauchy}}$ (results not shown).

5 Discussion

We presented a new, general rule for covariance matrix updates in variable metric real-valued direct search algorithms. The update rule operates directly on *a factorization* of the

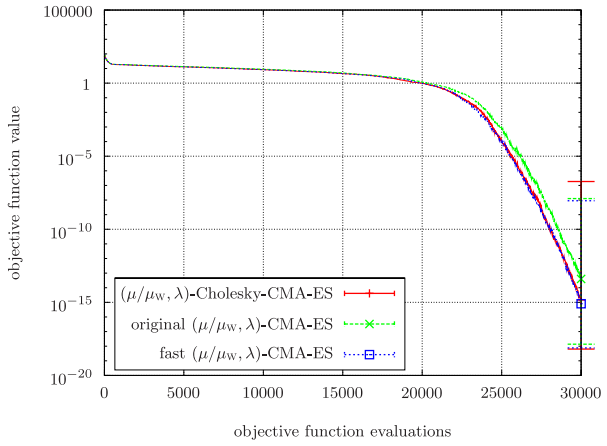

Fig. 7 Comparison of the $(\mu/\mu_W, \lambda)$-CMA-ES with rank-one updating and the simplified $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES on the benchmark function $f_{\text{Rosenbrock}}$ for $n = 20$. The $(\mu/\mu_W, \lambda)$-CMA-ES with original and increased learning rate c_σ are denoted by *original* and *fast* $(\mu/\mu_W, \lambda)$-CMA-ES, respectively. The trajectories show the medians of 25 trials, the final error bars the 10th and 90th percentile. The curves of fast $(\mu/\mu_W, \lambda)$-CMA-ES and $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES can hardly be distinguished

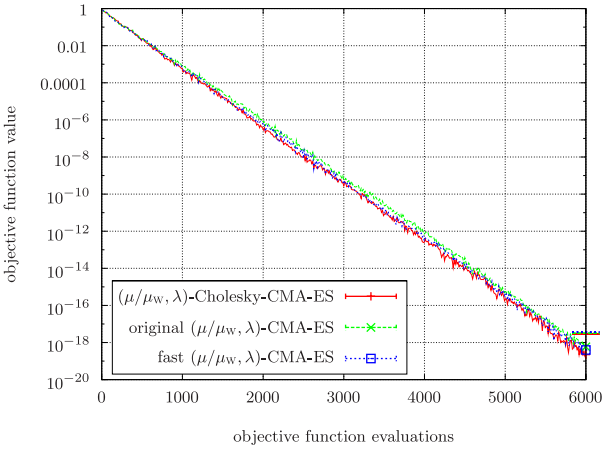

Fig. 8 Comparison of the $(\mu/\mu_W, \lambda)$-CMA-ES with rank-one updating and the simplified $(\mu/\mu_W, \lambda)$-Cholesky-CMA-ES on the noisy benchmark function $f_{\text{sphereCauchy}}$ for $n = 20$. The $(\mu/\mu_W, \lambda)$-CMA-ES with original and increased learning rate c_σ are denoted by *original* and *fast* $(\mu/\mu_W, \lambda)$-CMA-ES, respectively. The trajectories show the medians of 25 trials, the final error bars the 10th and 90th percentile. The three curves can hardly be distinguished

covariance matrix making the usually needed, repeated decompositions of the covariance matrix unnecessary. By simultaneously maintaining the inverse of the factors, the rule can replace *arbitrary* rank-one updates. This brings the following advantages:

- For n-dimensional problems, the new update requires $\Theta(n^2)$ computations and therefore reaches the asymptotically computational lower bound for the matrix update.
- We verified experimentally that the proposed matrix update is numerically stable and efficient in practice. For search space dimensions between $n = 10$ and 1000 the update proves to be about $n/2$ times faster than the otherwise necessary Cholesky decomposition.
- The proposed update is completely specified without hidden or numerically involved procedures such as an eigenvalue decomposition. Therefore, implementations in low level programming languages and even in hardware become easily possible.

The new learning rule can replace the original rank-one update of the covariance matrix and the subsequent matrix decomposition in the single- and multi-objective CMA-ES:

- It can be used with the elitist $(1 + \lambda)$-CMA-ES (Igel et al. 2006, 2007a) and the MO-CMA-ES (Igel et al. 2007a, 2007b) without changing the algorithms. As opposed to pre-

viously introduced incremental update rules (Igel et al. 2006, 2007b), restricted updates (Ostermeier 1992; Hansen et al. 1995; Poland and Zell 2001; Knight and Lunacek 2007; Ros and Hansen 2008), or to using outdated distributions (Hansen and Ostermeier 1996, 2001), here an efficient $\Theta(n^2)$ update is achieved without loss in performance.
– We combined the new update procedure with a slightly simplified non-elitist CMA-ES. The new algorithm performs in our experiments on par with the standard $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES with rank-one update (Hansen and Ostermeier 2001), but lacks its geometrical interpretation of conjugate perpendicularity.

The matrix update can be straightforwardly extended to rank-μ updating of the covariance matrix (Hansen et al. 2003; Hansen and Kern 2004) leading to an algorithm with time complexity $\Omega(\mu n^2)$. However, whether the rank-μ update can be conducted more efficiently than outlined above, whether the $(\mu/\mu_{\mathrm{w}}, \lambda)$-Cholesky-CMA-ES works well with larger values for μ, and detailed investigations of rank-μ algorithms remain subjects for future work.

Two drawbacks of our new method can be recognized:

– In combination with the non-elitist CMA-ES the original algorithm needs to be slightly modified. This drawback can be rectified with a symmetrical factorization of the covariance matrix. Therefore, the question remains to be addressed in future whether an efficient matrix update can be found where the Cholesky factor is symmetrical.
– The eigenvalues of the search distribution are not directly available for inspection. However, the eigenspectrum gives insight into the structure of the underlying problem. But because rare additional computations of the eigenspectrum for inspection do not have any effect on the algorithm performance, this drawback is of minor relevance.

Concluding our discussion, we believe that the easier implementation and the improved performance for large scale problems will make the CMA-ES more popular in particular in the machine learning community.

6 Conclusions

Evolution strategies (ESs) with variable metric are powerful tools in machine learning for search and optimization in continuous domains. They sample new candidate solutions according to a multi-variate normal distribution and adapt the covariance matrix to allow for efficient search even when the problem is non-separable and ill-conditioned. We proposed an incremental learning rule for the covariance matrix. We proved that it can equally replace the rank-one update together with the Cholesky decomposition of the covariance matrix, for example implemented in elitist single- and multi-objective covariance matrix adaptation ESs (CMA-ESs). The new learning rule reduces the computational complexity for a rank-one update of the search distribution to $\Theta(n^2)$, the asymptotically tight bound. The new method is considerably easier to implement, considerably faster in large dimensions and provides a significant improvement for high dimensional optimization and machine learning problems with fast computable performance measures.

Acknowledgements We thank the editor and anonymous reviewers for their highly appreciated feedback.

References

Arnold, D. V. (2002). *Noisy optimization with evolution strategies*. Dordrecht: Kluwer Academic.

Arnold, D. V. (2006). Weighted multirecombination evolution strategies. *Theoretical Computer Science*, *361*(1), 18–37.
Arnold, D. V., & Beyer, H. G. (2003). A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, *24*(1), 135–159.
Auger, A. (2005). Convergence results for the (1, λ)-SA-ES using the theory of φ-irreducible Markov chains. *Theoretical Computer Science*, *334*(1), 35–69.
Auger, A., & Hansen, N. (2005a). Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2005)* (pp. 1777–1784). New York: IEEE Press.
Auger, A., & Hansen, N. (2005b). A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2005)* (pp. 1769–1776). New York: IEEE Press.
Beume, N., & Rudolph, G. (2006). Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. In *IASTED international conference on computational intelligence* (pp. 231–236). Calgary: ACTA Press.
Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, *181*(3), 1653–1669.
Beyer, H. G. (2001). *The theory of evolution strategies*. New York: Springer.
Beyer, H. G. (2007). Evolution strategies. *Scholarpedia*, *2*(8), 1965.
Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies: A comprehensive introduction. *Natural Computing*, *1*(1), 3–52.
Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. New York: Wiley.
Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.
Emmerich, M., Beume, N., & Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. In C. A. Coello Coello, E. Zitzler, & A. Hernandez Aguirre (Eds.), *LNCS*: *Vol. 3410. Third international conference on evolutionary multi-criterion optimization (EMO 2005)* (pp. 62–76). New York: Springer.
Fogel, D. B. (1995). *Evolutionary computation: Toward a new philosophy of machine intelligence*. New York: IEEE Press.
Fonseca, C. M., Paquete, L., & López-Ibáñez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In: *IEEE congress on evolutionary computation* (pp. 1157–1163).
Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, *64*(C), 107–117.
Glasmachers, T., & Igel, C. (2008). Uncertainty handling in model selection for support vector machines. In G. Rudolph (Ed.), *LNCS*: *Vol. 5199. Parallel problem solving from nature (PPSN X)* (pp. 185–194). New York: Springer.
Gomez, F., Schmidhuber, J., & Miikkulainen, R. (2006). Efficient non-linear control through neuroevolution. *LNCS*: *Vol. 4212*. In *Proceedings of the European conference on machine learning (ECML 2006)* (pp. 654–662). New York: Springer.
Grewal, M. S., & Andrews, A. P. (1993). *Kalman filtering: Theory and practice*. Englewood Cliffs: Prentice-Hall.
Hager, W. W. (1989). Updating the inverse of a matrix. *SIAM Review*, *31*(2), 221–239.
Hansen, N. (1998). *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Eine Untersuchung zur entstochastisierten, koordinatensystemunabhängigen Adaptation der Mutationsverteilung*. Berlin: Mensch und Buch Verlag. ISBN 3-933346-29-0.
Hansen, N. (2006). The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, & E. Bengoetxea (Eds.), *Towards a new evolutionary computation. Advances on estimation of distribution algorithms* (pp. 75–102). New York: Springer.
Hansen, N. (2008). http://www.bionik.tu-berlin.de/user/niko/cmaapplications.pdf.
Hansen, N., & Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, & H. P. Schwefel (Eds.), *LNCS*: *Vol. 3242. Parallel problem solving from nature (PPSN VIII)* (pp. 282–291). New York: Springer.
Hansen, N., & Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the 1996 IEEE conference on evolutionary computation (ICEC '96)* (pp. 312–317).
Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.
Hansen, N., Ostermeier, A., & Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. Eshelman (Ed.), *Proceedings of the sixth international conference on genetic algorithms* (pp. 57–64). San Francisco: Morgan Kaufmann.

Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, *11*(1), 1–18.

Hansen, N., Niederberger, S. P. N., Guzzella, L., & Koumoutsakos, P. (2008). A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*. doi:10.1109/TEVC.2008.924423.

Heidrich-Meisner, V., & Igel, C. (2008a). Evolution strategies for direct policy search. In G. Rudolph (Ed.), *LNCS*: *Vol. 5199. Parallel problem solving from nature (PPSN X)* (pp. 428–437). New York: Springer.

Heidrich-Meisner, V., & Igel, C. (2008b). Similarities and differences between policy gradient methods and evolution strategies. In: M. Verleysen (Ed.), *16th European symposium on artificial neural networks (ESANN 2008)*, Evere, Belgien: d-side publications (pp. 427–432).

Heidrich-Meisner, V., & Igel, C. (2008c). Variable metric reinforcement learning methods applied to the noisy mountain car problem. In S. Girgin et al. (Eds.), *LNAI*: *Vol. 5323. European workshop on reinforcement learning (EWRL 2008)* (pp. 136–150). New York: Springer.

Igel, C. (2003). Neuroevolution for reinforcement learning using evolution strategies. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, & T. Gedeon (Eds.), *Vol. 4. Congress on evolutionary computation (CEC 2003)* (pp. 2588–2595). New York: IEEE Press.

Igel, C. (2005). Multi-objective model selection for support vector machines. In C. A. Coello Coello, E. Zitzler, & A. Hernandez Aguirre (Eds.), *LNAI*: *Vol. 3410. Third international conference on evolutionary multi-criterion optimization (EMO 2005)* (pp. 534–546). New York: Springer.

Igel, C., & Toussaint, M. (2003). On classes of functions for which No Free Lunch results hold. *Information Processing Letters*, *86*(6), 317–321.

Igel, C., Erlhagen, W., & Jancke, D. (2001). Optimization of dynamic neural fields. *Neurocomputing*, *36*(1–4), 225–233.

Igel, C., Suttorp, T., & Hansen, N. (2006). A computational efficient covariance matrix update and a $(1 + 1)$-CMA for evolution strategies. In *Proceedings of the genetic and evolutionary computation conference (GECCO 2006)* (pp. 453–460). New York: ACM Press.

Igel, C., Hansen, N., & Roth, S. (2007a). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, *15*(1), 1–28.

Igel, C., Suttorp, T., & Hansen, N. (2007b). Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, & T. Murata (Eds.), *LNCS*: *Vol. 4403. Fourth international conference on evolutionary multi-criterion optimization (EMO 2007)* (pp. 171–185). New York: Springer.

Igel, C., Glasmachers, T., & Heidrich-Meisner, V. (2008). Shark. *Journal of Machine Learning Research*, *9*, 993–996.

Iorio, A., & Li, X. (2005). Solving rotated multi-objective optimization problems using differential evolution. In G. I. Webb & X. Yu (Eds.), *LNCS*: *Vol. 3339. Proceeding of the 17th joint Australian conference on artificial intelligence* (pp. 861–872). New York: Springer.

Jägersküpper, J. (2006). How the $(1 + 1)$ ES using isotropic mutations minimizes positive definite quadratic forms. *Theoretical Computer Science*, *36*(1), 38–56.

Jägersküpper, J. (2007). Lower bounds for hit-and-run direct search. In *Proceedings of the fourth international symposium on stochastic algorithms: foundations and applications (SAGA)* (pp. 118–129). New York: Springer.

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

Jin, Y. (Ed.). (2006). *Multi-objective machine learning*. New York: Springer.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eshelman (Ed.), *Proceedings of the 6th international conference on genetic algorithms* (pp. 184–192). San Francisco: Morgan Kaufmann.

Kassahun, Y., & Sommer, G. (2005). Efficient reinforcement learning through evolutionary acquisition of neural topologies. In M. Verleysen (Ed.), *13th European symposium on artificial neural networks (ESANN 2005)* d-side (pp. 259–266).

Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., & Koumoutsakos, P. (2004). Learning probability distributions in continuous evolutionary algorithms—A comparative review. *Natural Computing*, *3*, 77–112.

Klee, V. (1977). Can the measure of $\bigcup[a_i, b_i]$ be computed in less than $O(n \cdot \log n)$ steps? *American Mathematical Monthly*, *84*, 284–285.

Knight, J. N., & Lunacek, M. (2007). Reducing the space-time complexity of the CMA-ES. In *Proceedings of the genetic and evolutionary computation conference (GECCO 2007)* (pp. 658–665). New York: ACM Press.

Mandischer, M. (2002). A comparison of evolution strategies and backpropagation for neural network training. *Neurocomputing*, *42*(1–4), 87–117.

Mersch, B., Glasmachers, T., Meinicke, P., & Igel, C. (2007). Evolutionary optimization of sequence kernels for detection of bacterial gene starts. *International Journal of Neural Systems*, *17*(5), 369–381, special issue on selected papers presented at the international conference on artificial neural networks (ICANN 2006).

Mierswa, I. (2006). Evolutionary learning with kernels: a generic solution for large margin problems. In *Proceedings of the 8th annual conference on genetic and evolutionary computation (GECCO 2006)* (pp. 1553–1560).

Mierswa, I. (2007). Controlling overfitting with multi-objective support vector machines. In *Proceedings of the 9th annual conference on genetic and evolutionary computation (GECCO 2007)* (pp. 1830–1837).

Miettinen, K. (1999). *Kluwer's international series in operations research & management science*: *Vol. 12*: *Nonlinear multiobjective optimization*. Dordrecht: Kluwer Academic.

Ostermeier, A. (1992). An evolution strategy with momentum adaptation of the random number distribution. *Parallel Problem Solving from Nature*, *2*, 197–206.

Overmars, M. H., & Yap, C. K. (1991). New upper bounds in Klee's measure problem. *SIAM Journal on Computing*, *20*(6), 1034–1045.

Pellecchia, A., Igel, C., Edelbrunner, J., & Schöner, G. (2005). Making driver modeling attractive. *IEEE Intelligent Systems*, *20*(2), 8–12.

Poland, J., & Zell, A. (2001). Main vector adaptation: A CMA variant with linear time and space complexity. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, & E. Burke (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO 2001)* (pp. 1050–1055). San Francisco: Morgan Kaufmann.

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

Ros, R., & Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In G. Rudolph (Ed.), *LNCS*: *Vol. 5199*. *Parallel problem solving from nature (PPSN X)* (pp. 296–305). New York: Springer.

Rudolph, G. (1992). On correlated mutations in evolution strategies. In R. Männer & B. Manderick (Eds.), *Parallel problem solving from nature 2 (PPSN II)* (pp. 105–114). Amsterdam: Elsevier.

Runarsson, T. P., & Sigurdsson, S. (2004). Asynchronous parallel evolutionary model selection for support vector machines. *Neural Information Processing—Letters and Reviews*, *3*(3), 59–68.

Schneider, S., Igel, C., Klaes, C., Dinse, H., & Wiemer, J. (2004). Evolutionary adaptation of nonlinear dynamical systems in computational neuroscience. *Journal of Genetic Programming and Evolvable Machines*, *5*(2), 215–227.

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

Schwefel, H. P. (1995). *Sixth-generation computer technology series: Evolution and optimum seeking*. New York: Wiley.

Siebel, N. T., & Sommer, G. (2007). Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, *4*(3), 171–183.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An Introduction*. Cambridge: MIT Press.

Suttorp, T., & Igel, C. (2006). Multi-objective optimization of support vector machines. In Y. Jin (Ed.), *Studies in computational intelligence*: *Vol. 16*. *Multi-objective machine learning* (pp. 199–220). New York: Springer.

Voß, T., Beume, N., Rudolph, G., & Igel, C. (2008). Scalarization versus indicator-based selection in multi-objective CMA evolution strategies. In *IEEE congress on evolutionary computation 2008 (CEC 2008)* (pp. 3041–3048). New York: IEEE Press.

Whitley, D., Lunacek, M., & Sokolov, A. (2006). Comparing the niches of CMA-ES, CHC and pattern search using diverse benchmarks. *LNCS*: *Vol. 4193*. In *Parallel problem solving from nature (PPSN IX)* (pp. 988–997). New York: Springer.

Winter, S., Brendel, B., Pechlivanis, I., Schmieder, K., & Igel, C. (2008). Registration of CT and intraoperative 3D ultrasound images of the spine using evolutionary and gradient-based methods. *IEEE Transactions on Evolutionary Computation*, *12*(3), 284–296.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, & H. P. Schwefel (Eds.), *Parallel problem solving from nature (PPSN V)* (pp. 292–301). New York: Springer.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132.