Adaptive Encoding: How to Render Search Coordinate System Invariant

Nikolaus Hansen

Microsoft Research–INRIA Joint Centre, 28 rue Jean Rostand, 91893 Orsay Cedex, France
nikolaus.hansen@inria.fr

Abstract. This paper describes a method for rendering search coordinate system independent, *Adaptive Encoding*. Adaptive Encoding is applicable to any iterative search algorithm and employs incremental changes of the representation of solutions. One attractive way to change the representation in the continuous domain is derived from Covariance Matrix Adaptation (CMA). In this case, adaptive encoding recovers the CMA evolution strategy, when applied to an evolution strategy with cumulative step-size control. Consequently, adaptive encoding provides the means to apply CMA-like representation changes to any search algorithm in the continuous domain. Experimental results confirm the expectation that CMA-based adaptive encoding will generally speed up a typical evolutionary algorithm on non-separable, ill-conditioned problems by orders of magnitude.

1 Introduction

In optimization or search, the problem encoding, that is the choice of the representation of the optimization problem is of utmost importance. A good representation, if available, can render any search problem trivial—finding a proper representation means essentially solving the problem. In an iterative search procedure, in principle, a good problem representation can be iteratively approached, just as a good solution to the problem is approached in the iteration sequence. Indeed, variable metric methods like quasi-Newton methods [2], covariance matrix adaptation (CMA) [7], or estimation of distribution algorithms [8] *implicitly* conduct a representational change. In case of an additive modification of solutions, as for example a mutation in an evolutionary algorithm, a linear change of representation is equivalent with an appropriate linear transformation of the additive mutation [3]. Linear transformations of additive mutation operators, parameterized in step-sizes or covariance matrices, are well studied in evolutionary algorithms [1,8].

In this paper, we sketch an *explicit* framework for an iterative incremental representation change, denoted as *adaptive encoding*. The framework by itself is just about trivial. While the framework is very general, this paper considers subsequently only *linear* changes of the representation *in the continuous domain*.

Searching for a linear representational change in the continuous domain complements the original n-dimensional search problem with a second search problem of size n^2. The advantage from adaptive encoding is that these two search problems are decoupled. Consequently, an effective adaptation of the representation (which can dramatically improve the algorithms performance) can be applied to any underlying search procedure.

G. Rudolph et al. (Eds.): PPSN X, LNCS 5199, pp. 205–214, 2008.

In the next section we give the preliminary notations and definitions. In Section 3 the general idea of adaptive encoding is introduced. Section 4 proposes an update rule for the representation, AE$_{\mathrm{CMA}}$, derived from CMA. We can prove that AE$_{\mathrm{CMA}}$ applied to CSA-ES recovers CMA-ES. Section 5 conducts another experimental proof of concept and Section 6 gives a summary and conclusions.

2 Preliminary Notations and Definitions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an objective function to be minimized. Let a (baseline) search algorithm propose new candidate solutions, $\boldsymbol{x}$, in an iterated procedure and typically evaluate them on f. Let further denote

S the **state space** of the search algorithm;
$\mathcal{A} : S \to S$ an iteration step of the search algorithm $\mathcal{A}$;
$T_B : S \to S$ an invertible transformation, the **decoding** of the state space, the change of representation. The T_B is parameterized by a matrix $\boldsymbol{B}$ and therefore uniquely depends on $\boldsymbol{B}$;
$\boldsymbol{B} \in \mathbb{R}^{n\times n}$ a full rank matrix, representing (i) a new **coordinate system** and a coordinate system transformation in $\mathbb{R}^n$, and (ii) a problem *representation* and linear *decoding* of candidate solutions $\boldsymbol{B} : \boldsymbol{x} \mapsto \boldsymbol{B}\boldsymbol{x}$;
$\mathcal{U} : \mathbb{R}^{n\times n} \times S \to \mathbb{R}^{n\times n}, (\boldsymbol{B}, s) \mapsto \mathcal{U}(\boldsymbol{B}, s)$ the change of representation by updating the matrix $\boldsymbol{B}$. For convenience, we assume that all necessary information to update $\boldsymbol{B}$ is included in the algorithm state s and we may write $\mathcal{U}(\boldsymbol{B})$ instead of $\mathcal{U}(\boldsymbol{B}, s)$;

From these definitions we first remark, that an iteration step of an algorithm can be surrounded by an encoding-decoding step according to

$$\mathcal{A}_B \equiv T_B \circ \mathcal{A} \circ T_B^{-1} \quad , \tag{1}$$

defining algorithm $\mathcal{A}_B : S \to S$, $s \mapsto T_B(\mathcal{A}(T_B^{-1}(s)))$. If T_B is the identity we have $\mathcal{A}_B \equiv \mathcal{A}$.

By definition, *decoded* solutions (phenotypes) are represented in the *given* coordinate system, where also f is evaluated. Accordingly, the algorithm operates, by definition, on *encoded* solutions (genotypes). We usually assume, for convenience and w.l.o.g., that recently evaluated solutions are part of the algorithms state.

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

Considering Remark 1, we can execute the algorithm $\mathcal{A}$ in any coordinate system of our choice. The new coordinate system, where the operations of $\mathcal{A}$ are effectively conducted, is defined by $\boldsymbol{B}$. Optimizing $f \circ \boldsymbol{B}$ instead of f already renders $\mathcal{A}$ independent of the given coordinate system (if $\boldsymbol{B}$ is chosen independent of the given coordinate system). Eq. (1) becomes meaningful when we also adapt $\boldsymbol{B}$. We shall choose T_B such that changes of $\boldsymbol{B}$ do not change solutions after they are decoded to their original representation (phenotype).

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

3 Adaptive Encoding

Equation (1) represents an iteration step of a search algorithm with an additional encoding-decoding procedure. The encoding is, throughout this paper, parameterized by a $n \times n$-matrix; it therefore adds n^2 degrees of freedom. Obviously, the idea is to find a good encoding for algorithm $\mathcal{A}$.

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

$$T_B \circ \mathcal{A} \circ T_B^{-1} \textit{ outperforms } \mathcal{A} \textit{ on } f$$

The static encoding is usually part of the design of the objective function. Equivalently, the algorithm can be modified specifically in regard to the given objective function (the encoding-decoding can certainly be interpreted as part of the algorithm). The formalism of Aim 1 is not very interesting. To get a more interesting situation, we need to consider an *update* or *adaptation* of the encoding T_B.

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

$$s \leftarrow T_B \circ \mathcal{A} \circ T_B^{-1}(s) \tag{2}$$

$$\boldsymbol{B} \leftarrow \mathcal{U}(\boldsymbol{B}, s) \tag{3}$$

where $\leftarrow$ denotes the assignment operator and $T_B \circ \mathcal{A} \circ T_B^{-1}(s) = T_B(\mathcal{A}(T_B^{-1}(s)))$. We write $T_B \circ \mathcal{A} \circ T_B^{-1}\,;\, \mathcal{U}(T_B)$ to denote the iteration step of Equations (2) *and* (3).

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

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

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

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

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

Aim 2 (adaptive encoding). *Find an update $\mathcal{U}$, such that for a given T_0 and a given (initial) T_B.*

$$T_B \circ \mathcal{A} \circ T_B^{-1}\,;\, \mathcal{U}(T_B) \textit{ outperforms } T_0 \circ \mathcal{A} \circ T_0^{-1} \textit{ on } f.$$

The left iteration step updates the encoding, the right iteration step applies a constant encoding, T_0, to algorithm $\mathcal{A}$.

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

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

Scenario 1. *(Standard scenario) The initial T_B equals to T_0. Aim 2 shall be satisfied for most given T_0.*

Scenario 2. *(Ambitious scenario) The initial T_B equals to T_0. Aim 2 shall be satisfied* for all *given T_0.*

Satisfying the ambitious scenario implies that no fixed optimal encoding T_B exists and a changing encoding can, in principle, be better than any fixed encoding. Both, the standard and the ambitious scenario are reasonable objectives, depending on the given objective function.

The remainder of this paper proposes and investigates an effective way to implement adaptive encoding as given in Definition 1.

4 Adaptive Encoding Based on Covariance Matrix Adaptation

In order to define an adaptive encoding, we need to specify the encoding of the algorithms state space, $T_B : S \to S$, and the update of the encoding, $\mathcal{U}$. In this section, our aim is to obtain an efficient update $\mathcal{U}$, leaving the choice of T_B as only remaining, algorithm specific design issue. The update is derived from the equations for the covariance matrix update in the $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES [4,7], denoted as AE$_\mathrm{CMA}$ in the following, and explicated in Algorithm 1 AE$_\mathrm{CMA}$-Update.

The parameters of AE$_\mathrm{CMA}$-Update are chosen to $\alpha_0 = \frac{\sqrt{n}}{\|\boldsymbol{B}^{-1}(\boldsymbol{m}-\boldsymbol{m}^-)\|}$, with $l_i = \|\boldsymbol{B}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}^-)\|$ we have $\alpha_i = \sqrt{n}\,\max\Big(\frac{l_i}{\beta}, \operatorname*{median}_{j=1,\dots,\mu}(l_j)\Big)^{-1}$, for $i = 1, \dots, \mu$ and $\beta = 2$, $\alpha_\mathrm{p} = 1$, $c_\mathrm{p} = \frac{1}{\sqrt{n}}$, $w_i = \frac{\ln(\mu+1) - \ln i}{\mu\ln(\mu+1) - \sum_{j=1}^{\mu}\ln j}$, for $i = 1, \dots, \mu$, and μ is half of the overall generated number of solutions per iteration (before selection), $c_1 = \alpha_\mathrm{c}\frac{0.2}{(n+1.3)^2+\mu_\mathrm{w}}$, $c_\mu = \alpha_\mathrm{c}\frac{0.2\left(\mu_\mathrm{w}-2+\frac{1}{\mu_\mathrm{w}}\right)}{(n+2)^2+0.2\mu_\mathrm{w}}$ with $\mu_\mathrm{w} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$. Finally $\alpha_\mathrm{c} \approx 1$ must be chosen positive and such that $c_1 + c_\mu \leq 1$. Too large values for α_c potentially lead to a failure. Too small values slow down the adaptation. In any case, a parameter study for α_c is recommended, as conducted below. All parameters are detailed in [5].

The state variables are $\boldsymbol{m}$, $\boldsymbol{p}$ and $\boldsymbol{C}$. The mean $\boldsymbol{m}$ is initialized to the initial solution mean of the search algorithm to which AE$_\mathrm{CMA}$-Update is applied, and initially $\boldsymbol{p} = \boldsymbol{0}$ and $\boldsymbol{C} = \boldsymbol{I}$.

Proposition 2. *Let σ denote a step-size and $\mu_\mathrm{w}^{-1} = \sum_{i=1}^{\mu} w_i^2$. Let $\alpha_p = 1$, $\alpha_0 = \frac{\sqrt{\mu_\mathrm{w}}}{\sigma}$ and $\alpha_i = \sigma^{-1}$, for $i = 1, \dots, \mu$. Then, the procedure AE$_{CMA}$-Update implements*

Algorithm 1. AE$_{\text{CMA}}$-Update($\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\}$)
updates the encoding matrix $\boldsymbol{B}$ using the μ recent best-ranked candidate solutions

```
1  given parameters $w_i, c_{\text{p}}, c_1, c_\mu$, see text
2  given $\boldsymbol{m} \in \mathbb{R}^n$, $\boldsymbol{p} \in \mathbb{R}^n$ and $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ from last iteration
3  let matrices $\boldsymbol{B}^\circ$ orthogonal, and $\boldsymbol{D}$ diagonal, with diagonal elements sorted in ascending
   order , "$\leftarrow$" assigns accordingly
4  $\boldsymbol{m}^- = \boldsymbol{m}$
5  $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_i$                                  // Eq.(3) in [4]
6  set scalars $\alpha_i \geq 0$, for $i = 0, \ldots, \mu$, see text
7  $\boldsymbol{p} \leftarrow (1 - c_{\text{p}})\,\boldsymbol{p} + \sqrt{c_{\text{p}}\,(2 - c_{\text{p}})}\,\alpha_0(\boldsymbol{m} - \boldsymbol{m}^-)$          // Eq.(17) in [4]
8  $\boldsymbol{C}_\mu = \sum_{i=1}^{\mu} w_i\, \alpha_i^2\, (\boldsymbol{x}_i - \boldsymbol{m}^-)(\boldsymbol{x}_i - \boldsymbol{m}^-)^{\mathrm{T}}$           // rank-$\mu$ matrix
9  set scalar $\alpha_{\text{p}} \geq 0$, see text
10 $\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu)\,\boldsymbol{C} + c_1\alpha_{\text{p}}\,\boldsymbol{p}\boldsymbol{p}^{\mathrm{T}} + c_\mu \boldsymbol{C}_\mu$               // Eq.(22) in [4]
11 $\boldsymbol{B}^\circ\boldsymbol{D}\boldsymbol{D}\boldsymbol{B}^\circ \leftarrow \boldsymbol{C}$                                 // eigendecomposition
12 optionally normalize $\boldsymbol{D}$
13 $\boldsymbol{B} \leftarrow \boldsymbol{B}^\circ\boldsymbol{D}$                                       // encoding matrix
```

the update equations for the evolution path, $\mathbf{p}$, and the covariance matrix, $\boldsymbol{C}$, in the ($\mu/\mu_{\text{W}}, \lambda$)-CMA-ES.

Proof. Assuming that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu$ are the μ best solutions in the recent iteration step, line 5 computes $\boldsymbol{m}$ according to Eq. (3) in [4]. Lines 7 and 10 of AE$_{\text{CMA}}$-Update replicate the covariance matrix update equations (17) and (22) in [4] with added or renamed normalization coefficients, denoted with α. Substituting the coefficients as given above results in the original equations.

The AE$_{\text{CMA}}$-Update implements the covariance matrix update of CMA-ES with additional coefficients α to be specified. Within CMA-ES, this update was designed to operate reliably for any choice of μ [4].

Depending on the application of AE$_{\text{CMA}}$-Update, a slow change of $\boldsymbol{B}$ might be desirable. While $\boldsymbol{C}$ will only change slowly, as long as c_1 and c_μ are small, the decomposition of $\boldsymbol{C}$ does not ensure a similar behavior for $\boldsymbol{B}^\circ$ and $\boldsymbol{D}$. For this reason, the diagonal elements in $\boldsymbol{D}$ are sorted. As an approximation, it might even be sufficient to only decode the solutions for the function evaluation and completely abandon the encoding-decoding of the algorithms state.

AE$_{CMA}$ Recovers CMA-ES. We apply AE$_{\text{CMA}}$ (Algorithm 1) to an evolution strategy with cumulative step-size adaptation (CSA, sometimes also denoted as *path length control*). The AE$_{\text{CMA}}$-($\mu/\mu_{\text{W}}, \lambda$)-CSA-ES is given in Algorithm 2, where the begin-end block marks the original ($\mu/\mu_{\text{W}}, \lambda$)-CSA-ES. The following invertible encoding for the state variables in CSA-ES is used.

$$T_B : (\boldsymbol{m}, \boldsymbol{p}_\sigma, \sigma) \mapsto (\boldsymbol{B}\boldsymbol{m}, \boldsymbol{B}^\circ\boldsymbol{p}_\sigma, \sigma) \tag{4}$$

Additionally, the μ best solutions are used as input to AE$_{\text{CMA}}$-Update (line 16 in Algorithm 2). The encoding T_B solely depends on $\boldsymbol{B}$, as $\boldsymbol{B}^\circ$ can be computed from $\boldsymbol{B}$ by normalizing its columns to one. Applying AE$_{\text{CMA}}$-Update to CSA-ES we find.

Algorithm 2. AE$_{\text{CMA}}$-CSA-ES

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

$i(f)$ indicates the index of the i-th best solution, e.g., $\boldsymbol{x}_{1(f)} = \arg\min_{i=1,\dots,\lambda} \{f(\boldsymbol{x}_i)\}$

Shaded areas implement the adaptive encoding, AE$_{\text{CMA}}$, also updating $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$

The begin-end block embraces the original CSA-ES minimizing $f \circ \boldsymbol{B}$

```
initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean), $\boldsymbol{p}_\sigma = \mathbf{0}$ (evolution path), $\sigma > 0$ (step-size)
initialize $\boldsymbol{B} = \boldsymbol{B}^\circ = \boldsymbol{I}$ (encoding matrices)
repeat
    $\boldsymbol{m} \leftarrow \boldsymbol{B}^{-1}\boldsymbol{m}$
    $\boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^{\circ\mathrm{T}}\boldsymbol{p}_\sigma$
    begin
        $\boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \boldsymbol{I})\,, \quad$ for $i = 1, \dots, \lambda$
        $f_i = f \circ \boldsymbol{B}\,(\boldsymbol{x}_i) = f(\boldsymbol{B}\,\boldsymbol{x}_i)\,,$ for $i = 1, \dots, \lambda$    // decode to evaluate
        $\boldsymbol{m}^- = \boldsymbol{m}$
        $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i(f)}$
        $\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2 - c_\sigma)\mu_{\mathrm{w}}}\,\frac{1}{\sigma}(\boldsymbol{m} - \boldsymbol{m}^-)$
        $\sigma \leftarrow \sigma\,\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|} - 1\right)\right)$
    end
    $\boldsymbol{m} \leftarrow \boldsymbol{B}\boldsymbol{m}$
    $\boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^\circ\boldsymbol{p}_\sigma$
    $\boldsymbol{B}, \boldsymbol{B}^\circ \leftarrow$ AE$_{\text{CMA}}$-Update$(\{\boldsymbol{B}\boldsymbol{x}_1, \dots, \boldsymbol{B}\boldsymbol{x}_\mu\})$    // update $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$
until stopping criterion is met
```

Theorem 1 (Recovery of CMA-ES). *Let T_B given in Eq.* (4) *and the scalars for AE$_{CMA}$-Update in each iteration given in Proposition 2, then the AE$_{CMA}$-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES (Algorithm 2) implements the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES.*

Proof. Due to the space limitations, the proof is provided in [5].

Theorem 1 supports the hypothesis that AE$_{\text{CMA}}$-Update is an effective way to update the representation matrix $\boldsymbol{B}$ in evolutionary search algorithms, as CMA-ES efficiently adapts the principle axes of the coordinate system, where the independent sampling takes place. In the next section, another application of AE$_{\text{CMA}}$-Update is realized.

5 Yet Another Experimental Proof of Concept

While AE$_{\text{CMA}}$-Update has proved to be effective with CSA-ES [7], in this section we provide another case study. To underline the general applicability of AE$_{\text{CMA}}$-Update we consider a baseline algorithm that (i) exploits the given coordinate system, and (ii) generates distributions that are rather different from Gaussians, providing a test scenario that is rather different from CSA-ES. We use a simple but functional algorithm that utilizes a Cauchy distribution in a derandomized adaptation framework, denoted as $(1, \lambda)$-Cauchy-ES, with $\lambda = 10$ (see the begin-end block in Algorithm 3). Despite this choice

Algorithm 3. AE$_{\text{CMA}}$-$(1,\lambda)$-Cauchy-ES
The begin-end block embraces the $(1,\lambda)$-Cauchy-ES minimizing $f \circ \boldsymbol{B}$
Shaded areas implement the adaptive encoding
$\boldsymbol{R}_i \in \mathbb{R}^n$ is standard Cauchy distributed in each component
$1(f)$ indicates the index of the best solution $\boldsymbol{x}_{1(f)} = \arg\min_{i=1,\ldots,\lambda}\{f(\boldsymbol{x}_i)\}$

```
initialize $\boldsymbol{x} \in \mathbb{R}^n$ and diagonal matrix $\sigma$ (step-size matrix)
initialize $\boldsymbol{B} = \boldsymbol{I}$ (encoding matrix)
repeat
    $\boldsymbol{x} \leftarrow \boldsymbol{B}^{-1}\boldsymbol{x}$
    begin
        $\boldsymbol{x}_i = \boldsymbol{x} + \sigma \boldsymbol{R}_i$,    for $i = 1, \ldots, \lambda$
        $f_i = f \circ \boldsymbol{B}(\boldsymbol{x}_i) = f(\boldsymbol{B}\boldsymbol{x}_i)$,  for $i = 1, \ldots, \lambda$    // decode to evaluate
        $\boldsymbol{x} \leftarrow \boldsymbol{x}_{1(f)}$
        $\sigma_{jj} \leftarrow \sigma_{jj} \exp\left(\frac{1}{2\,n}\left(\frac{1}{2}\,\text{sign}(|R_{1(f),j}| - 0.9) + \text{sign}\sum_{i=1}^{n}\text{sign}(|R_{1(f),i}| - 1)\right)\right)$
    end
    $\boldsymbol{x} \leftarrow \boldsymbol{B}\boldsymbol{x}$
    $\boldsymbol{B} \leftarrow$ AE$_{\text{CMA}}$-Update$(\{\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu\})$    // update $\boldsymbol{B}$
until *stopping criterion is met*
```

(lead by simplicity and personal preference), neither comma-selection (non-elitism) nor derandomization nor a small population size are fundamental prerequisites for applying AE$_{\text{CMA}}$. The $(1,\lambda)$-Cauchy-ES samples new solutions without dependencies between variables in the given coordinate system, because σ is a diagonal matrix. Rendering the $(1,\lambda)$-Cauchy-ES coordinate system independent results in correlations between variables (even if $\sigma = \boldsymbol{I}$), because the Cauchy distribution is highly anisotropic [6]. The invertible encoding

$$T_B : (\boldsymbol{x}, \sigma) \mapsto (\boldsymbol{B}\boldsymbol{x}, \sigma) \tag{5}$$

is used, where the step-size matrix σ is not transformed. An appropriate mapping for a covariance matrix $\sigma^2 \mapsto \boldsymbol{B}\sigma^2\boldsymbol{B}^{\mathrm{T}}$ would not preserve the diagonal property, while arguably $T_B^{-1} : \boldsymbol{C} \mapsto \mathtt{diag}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{B})$ could be used. In contrast, using $\boldsymbol{B}\mathtt{diag}(\sigma)$ as mapping for only the diagonal of σ cannot be recommended, because very small diagonal entries can occur accidentally. Because σ is not encoded, it is important that changes of $\boldsymbol{B}$ remain modest.

Using $\boldsymbol{B}^\circ$ instead of $\boldsymbol{B}$ in Eq. (5) is a possible alternative and investigated below. In this case, the step-size matrix σ needs to learn the scaling that can be otherwise provided by the diagonal matrix $\boldsymbol{D}$ in Algorithm 1.

Test functions. Testing on a number of functions, we always found the expected effect from AE$_{\text{CMA}}$. Exemplarily, we show simulations on two quadratic test functions, falling into Scenario 1 from Section 3.

$$f_{\text{elli}}(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{6\,\frac{i-1}{n-1}} y_i^2 \quad \text{and} \quad f_{\text{cigtab}}(\boldsymbol{x}) = y_1^2 + 10^4 \sum_{i=2}^{n-1} y_i^2 + 10^8 y_n^2 \ , \tag{6}$$

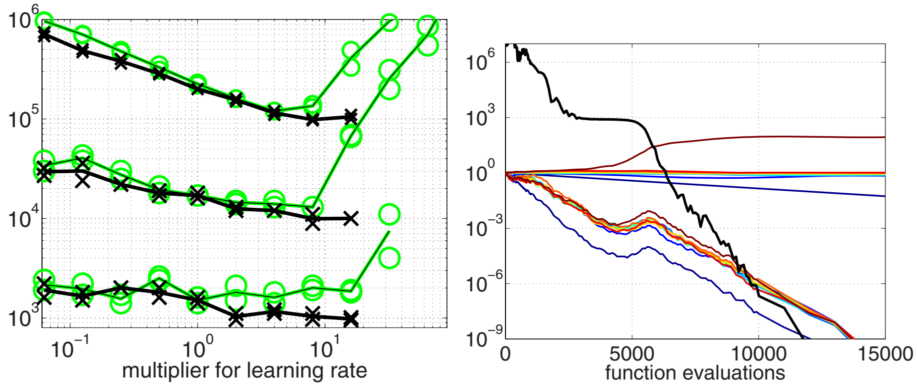

Fig. 1. Simulation of the AE$_{\mathrm{CMA}}$-$(1,\lambda)$-Cauchy-ES. Left: number of function evaluations to reach function value 10^{-9} on the rotated f_{elli} versus the multiplier α_{c} for the learning rate of $\boldsymbol{B}$ in 3-, 10- and 30-D (from bottom to top). Symbols $\times = \boldsymbol{B}$ and $\bigcirc = \boldsymbol{B}^{\circ}$. For each set-up two trials were conducted. Right: time evolution on the rotated f_{cigtab} in 10-D, shown are the objective function value (bold single graph), diagonal elements of the step-size matrix σ (lower group of curves) and diagonal elements of $\boldsymbol{D}$ in Algorithm 1 (smooth upper graphs).

where $\boldsymbol{y} := \boldsymbol{O}\boldsymbol{x}$ and $\boldsymbol{O} = [\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n]$ implements an angle-preserving, linear transformation, *i.e.* $\boldsymbol{O}$ is orthogonal. The basis $\boldsymbol{O}$ was either chosen as identity $\boldsymbol{I}$ (*axis-parallel* case), or each column was sampled uniformly distributed on the unit hypersphere, orthogonalized to the previous columns and normalized to one (*rotated* case). For each trial a new basis was sampled. Further initial values were $\boldsymbol{x} = (1, \ldots, 1)^{\mathrm{T}}$ and $\sigma = \boldsymbol{I}$. In the following, if a single trial is shown, it represents a typical trial.

Choosing parameters for AE$_{CMA}$. For applying AE$_{\mathrm{CMA}}$ to the $(1,\lambda)$-Cauchy-ES, we conduct a minimalistic parameter study for the multiplier, α_{c}, of the learning rate for the matrix $\boldsymbol{B}$. Further parameters for the AE$_{\mathrm{CMA}}$-Update follow the settings from Section 4, accordingly, we use $\mu = \lambda/2 = 5$. We test two cases, (i) using Eq. (5) and (ii) replacing $\boldsymbol{B}$ with $\boldsymbol{B}^{\circ}$ in Eq. (5). The remaining set-up is minimalistic. We test on the rotated f_{elli} in 3-, 10- and 30-D, vary α_{c} by factors of 2 and $1/2$ and measure the number of function evaluations to reach function value 10^{-9}, twice for each set-up. Results are shown in Figure 1, left. Missing points for large values of α_{c} (to the right) indicate that at least one run did not succeed. Large values lead to a failure, because the condition number of matrix $\boldsymbol{D}$ (line 11 in Algorithm 1) diverges. Using $\boldsymbol{B}^{\circ}$ is less prone to a failure. When reducing α_{c} to small values, the number of function evaluations will increase at most linearly with ${\alpha_{\mathrm{c}}}^{-1}$.

Only for large values of α_{c} the performance is remarkably different for $\boldsymbol{B}$ and $\boldsymbol{B}^{\circ}$. With increasing α_{c}, first $\boldsymbol{B}^{\circ}$ becomes worse, but finally $\boldsymbol{B}$ fails earlier than $\boldsymbol{B}^{\circ}$. Nevertheless, the designated default value $\alpha_{\mathrm{c}} = 1$ is applicable in both cases: the value is more than ten times larger than a value that leads to a failure and the performance loss to the best setting, does not exceed a factor of two. We retain using $\boldsymbol{B}$ as in Eq. (5) in the following.

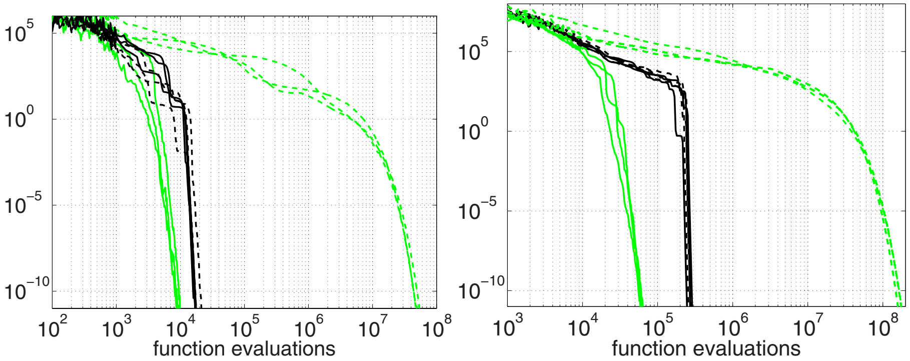

Fig. 2. Simulation of AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES (bold) and $(1, \lambda)$-Cauchy-ES (light) on the axis-parallel (solid) and the rotated (dashed) f_{elli} in 10-D (left) and 30-D (right). Shown is the objective function value of respectively 3 trials over time. In the rotated case, the AE$_{\text{CMA}}$ improves the performance by a factor of roughly one thousand.

Figure 1 (right) shows a single run on the rotated f_{cigtab} in 10-D. The function topology is successfully adapted as the optimum is approached quickly after about 5000 function evaluations. The single large dispersion value in the distribution, relating to y_1 in Eq. (6), is mainly represented in the matrix $\boldsymbol{D}$ (upper graph), while the single small value, relating to y_n, is mainly represented in σ, in particular in the early stage.

The comparison. Completing the picture, we compare the AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES with the $(1, \lambda)$-Cauchy-ES on f_{elli}. In Figure2, three runs are shown for each algorithm on the axis-parallel and on the rotated function in 10-D (left) and 30-D (right).

The performance of the AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES is virtually independent of rotation (only the initialization is still different in both cases). On the axis-parallel function, AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES becomes about two to ten times slower than $(1, \lambda)$-Cauchy-ES (for reaching function value 10^{-10} in 10-D and function value 100 in 30-D respectively). On the rotated function, AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES becomes between 200 and 2000 times(!) faster (for reaching function value 10^{-1} in 30-D and function value 10^{-10} in 10-D respectively). The application of the AE$_{\text{CMA}}$ was apparently successful (the CMA-ES is still roughly four times faster in this particular case). The trade-off when AE$_{\text{CMA}}$ is applied with the axis-parallel function is comparatively small, but increases with increasing dimension. By fixing the transformation $\boldsymbol{B}$ for some time in the beginning of the optimization, this trade-off can be eliminated.

6 Summary and Conclusions

We have outlined an *adaptive* change of representation in iterative search, denoted as *adaptive encoding* (AE): after each iteration step, (i) the algorithms state is "decoded", (ii) the encoding mechanism is adapted, and (iii) the algorithms state is "encoded" again for the next iteration. Additionally, candidate solutions are decoded for their evaluation on the objective function. In the continuous search domain, practical implications from this simple procedure are surprisingly far-reaching. The sophisticated update of the covariance matrix in the CMA-ES can be entirely formulated as a change of representation of an encoding matrix $\boldsymbol{B}$ (Proposition 2). Applying this representation change, $\mathrm{AE}_{\mathrm{CMA}}$, to an evolution strategy with cumulative step-size control, the CMA-ES is recovered (Theorem 1)—proving that an effective representation change can be entirely decoupled from the underlying search algorithm. Addressing an important open problem in evolutionary computation [9], the representation change implicitly induced by the covariance matrix adaptation in the CMA-ES becomes available for any continuous domain search algorithm—$\mathrm{AE}_{\mathrm{CMA}}$ *can render any search algorithm independent of the coordinate system*, in particular rotationally invariant.

We conjecture that on various non-separable ill-conditioned problems $\mathrm{AE}_{\mathrm{CMA}}$ will typically speed up population-based search methods by orders of magnitude. A case study of $\mathrm{AE}_{\mathrm{CMA}}$ supports our conjecture: the baseline algorithm has become roughly thousand times faster on the non-separable problems. While the principle of adaptive encoding is quite general, we anticipate successful applications (e.g. using Algorithm 1) in particular for population-based, stochastic search algorithms in the continuous domain.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
2. Davidon, W.: Variable Metric Method for Minimization. SIAM J. Optim. 1, 1 (1991)
3. Hansen, N.: Invariance, self-adaptation and correlated mutations in evolution strategies. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 355–364. Springer, Heidelberg (2000)
4. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a new evolutionary computation. Advances on estimation of distribution algorithms, pp. 75–102. Springer, Heidelberg (2006)
5. Hansen, N.: Adaptive encoding for optimization. Research Report 6518, INRIA (April 2008), `http://hal.inria.fr/inria-00275983/en`
6. Hansen, N., Gemperle, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 62–71. Springer, Heidelberg (2006)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
8. Larrañaga, P.: A review on estimation of distribution algorithms. In: Larrañaga, P., Lozano, J. (eds.) Estimation of distribution algorithms, pp. 80–90. Kluwer Academic Publishers, Dordrecht (2002)
9. Salomon, R.: Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. BioSystems 39(3), 263–278 (1996)