HAL
open science

Adaptive Encoding for Optimization

Nikolaus Hansen

To cite this version:

Nikolaus Hansen. Adaptive Encoding for Optimization. [Research Report] RR-6518, 2008. inria-00275983v2

HAL Id: inria-00275983

https://inria.hal.science/inria-00275983v2

Submitted on 27 Apr 2008 (v2), last revised 29 Apr 2008 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Adaptive Encoding for Optimization

Nikolaus Hansen

N° ????

Avril 2008

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche SACLAY - ÎLE-DE-FRANCE

Adaptive Encoding for Optimization

Nikolaus Hansen*

Thème COG — Systèmes cognitifs
Équipe-Projet TAO

Rapport de recherche n° ???? — Avril 2008 — 18 pages

Abstract: This report describes a general method for rendering search coordinate system independent, Adaptive Encoding (AE). Adaptive Encoding is applicable to any continuous domain search algorithm and includes (incremental) changes of the coordinate system, that is, changes of the representation of solutions. One attractive way to change the representation within AE is derived from the Covariance Matrix Adaptation (CMA). We prove that adaptive encoding recovers the CMA Evolution Strategy, when suitably applied to an evolution strategy with cumulative step-size control. The proof implies that adaptive encoding provides the means to apply CMA-like representation changes to any search algorithm in continuous domain.

Key-words: optimization, evolutionary algorithms, encoding, representation, coordinate system, covariance matrix adaptation, invariance

* INRIA Saclay, Project-team TAO, Université Paris-Sud, LRI, 91405 Orsay Cedex, France.
email:`forename.name@inria.fr`.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Encodage adaptif pour optimisation

Résumé : Pas de résumé

Mots-clés : Pas de motclef

Contents

1 Introduction **3**

2 Preliminary Notations and Definitions **4**
Symbols 5

3 Adaptive Encoding **5**

4 Example Applications of Adaptive Encoding **7**
4.1 The Scaling Adaptive Evolutionary Algorithm 7
4.2 EDA (Estimation of Distribution Algorithm) 9

5 A Universal Update Rule: $\mathbf{AE_{CMA}}$ **11**
5.1 Choice of Parameters 12
5.2 $\mathrm{AE_{CMA}}$ Recovers CMA-ES 14
5.3 Application of $\mathrm{AE_{CMA}}$-Update 17

6 Summary and Conclusions **17**

1 Introduction

In optimization or search, the problem encoding, that is the choice of the representation of the optimization problem is of upmost 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, for example, variable metric methods like quasi-Newton methods [3, 4], covariance matrix adaptation (CMA) [8], or estimation of distribution algorithms (EDAs) [9] implicitely conduct a representational change. In evolutionary algorithms, given an additive (mutation) operator, a linear change of representation is equivalent with an appropriate linear transformation of the additive mutation [5]. Linear transformations of additive mutation operators, parameterized in step-sizes or covariance matrices, are well studied in evolutionary algorithms [1, 9].

In this report, we sketch an explicite framework for an iterative incremental representation change, adaptive encoding. The framework by itself is just about trivial. While the framework is quite general, this report considers only linear changes of the representation in continuous domain.

Searching for a linear representational change in 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, any efficient representation change can be applied to any underlying search procedure.

In the next section we give the preliminary notations and definitions. In Section 3 the general idea of adaptive encoding is introduced. Section 4 explicates a few examples. In one of them adaptive encoding recovers an EDA. Section 5 proposes a universal, efficient update rule, $\mathrm{AE_{CMA}}$, for the representation. We prove that $\mathrm{AE_{CMA}}$-CSA-ES recovers CMA-ES.

2 Preliminary Notations and Definitions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an objective function to be minimized. Let a search algorithm propose new candidate solutions, $\boldsymbol{x}$, in an iterative 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 **encoding** of the state space or change of representation. The transformation T_B is here parameterized by a matrix $\boldsymbol{B}$ and uniquely depends on the values in $\boldsymbol{B}$. The inverse T_B^{-1} denotes the decoding of the state space;

$\boldsymbol{B} \in \mathbb{R}^{n\times n}$ a full rank matrix. Columns of $\boldsymbol{B}$ represent a new **coordinate system** and $\boldsymbol{B}$ represents the respective a coordinate system transformation in $\mathbb{R}^n$;

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

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

defining a new algorithm $\mathcal{A}_B : S \to S$. If T_B is the identity the original algorithm $\mathcal{A}$ is recovered.

By definition, encoded solutions (phenotypes) are represented in the given coordinate system, where also f is evaluated. Accordingly, the algorithm operates, by definition, on decoded solutions (genotypes). We 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 candidated solutions are evaluated according to their original representation. Solutions must be encoded before evaluation. In other words,* $\mathcal{A}$ *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 conducted, is given by $\boldsymbol{B}$. We can additionally adapt $\boldsymbol{B}$ (which completely determines T_B). In this way, Eq. (1) can render the algorithm $\mathcal{A}_B$ independent from any given coordinate system (if the adaptation for $\boldsymbol{B}$ is formulated independent of the coordinate system).

Finally, we assume to have a performance measure when running an algorithm on a function.

Remark 2 (Performance measure)**.** *In the following, we assume to have a performance measure for running a search algorithm on the 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 values is reached.

Further symbols which will be used repeatedly in this report are given in the next paragraph.

Symbols

$\circ$ the composition of mappings, e.g., $(f \circ \boldsymbol{B})(\boldsymbol{x}) = f(\boldsymbol{B}(\boldsymbol{x}))$ for all $\boldsymbol{x}$.

$\leftarrow$ the assignment operation. We will write, for example,

$$\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} \leftarrow \boldsymbol{C} \ ,$$

meaning we assign $\boldsymbol{B}$ to a value such that $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{C}$. If the asignment is not unique under the definition for $\boldsymbol{B}$, any compliant value is assigned.

μ, λ integers denoting the number of parental solutions and offspring solutions respectively.

$\mu_{\mathbf{w}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$ the effective parent number, also denoted as variance effective selection mass [6].

$\boldsymbol{I}$ the identity $n \times n$ unit matrix.

$i(f)$ the index of the i-the best solution in respect to function value f.

$\mathcal{N}_i(\mathbf{0}, \boldsymbol{C})$ a multi-variate normal distribution with zero mean and covariance matrix $\boldsymbol{C}$. The index i denotes a certain realization.

$R_i \in \mathbb{R}^n$ a random vector.

$\boldsymbol{B}^{\mathrm{T}}$ the transpose of $\boldsymbol{B}$.

$w_i = \frac{\ln(\mu+1) - \ln i}{\mu \ln(\mu+1) - \sum_{j=1}^{\mu} \ln j}$, for $i = 1, \ldots, \mu$, positive weights summing to one.

3 Adaptive Encoding

Equation (1) represents an iteration step of a search algorithm with an additional encoding-decoding procedure. The encoding is, throughout this report, parameterized by a $n \times n$-matrix; it therefore introduces n^2 additional degrees of freedom to the iteration step of the algorithm. 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 modification 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, determined by $\boldsymbol{B}$, and an update, $\mathcal{U}$, the iteration step of an adaptively encoded algorithm in state $s \in S$ is defined as*

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

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

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

$$T_B \circ \mathcal{A} \circ T_B^{-1}\,;\, \mathcal{U}(T_B) \tag{4}$$

to denote the iteration step defined by 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 an effective adaptive encoding update rule.

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*

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

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 <u>sequence</u> is considered. Indeed, in a realistic automated scenario, Aim 2 can only be achieved in the iteration sequence.

Finally, we define three 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 <u>for all</u> 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.

Scenario 3. *(Unrealistic scenario) For all $T_B \neq T_0$, Aim 2 shall be satisfied.*

As we might be able to choose T_B arbitrarily bad, it seems unrealistic that Aim 2 can be satisfied for any $T_B \neq T_0$ in general.

The remainder of this report further investigates the idea of adaptive encoding as given in Definition 1.

4 Example Applications of Adaptive Encoding

In this section, we give two examples of how an evolutionary algorithm is reformulated as a simple base algorithm with adaptive encoding.

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, $\mathcal{U}$. Both mappings depend, to a certain extend, on the baseline algorithm they are applied to.

4.1 The Scaling Adaptive Evolutionary Algorithm

We define in Algorithm 1 a "Baseline (μ, λ)-EA" with μ parental solutions, a recombination operator `reco`, λ offspring which are generated by adding a random vector $R_i \in \mathbb{R}^n$, and a selection operator. The (μ, λ)-selection was chosen for notational simplicity. The results generalize to any selection operator that uses only the objective function values to take decisions (and not the parameter values of the candidate solutions itself).

The adaptive encoding of the baseline (μ, λ)-EA, AE-(μ, λ)-EA, is given in Algorithm 2. It adds a decoding step in the beginning, encodes solutions for evaluation, adds an encoding step at the end and finally updates the encoding.

Remark 3. *The encoding T_B in the adaptively encoded (μ, λ)-EA (Algorithm 2) is of the form*

$$T_B : (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu, \sigma, y) \mapsto (\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu, \sigma, y) \ , \tag{5}$$

where y denotes remaining state variables.

We aim at investigating the relation of the adaptive encoding of the baseline (μ, λ)-EA to conventional adaptations in evolutionary algorithms. Algorithm 3 complements the baseline (μ, λ)-EA with an adaptive scaling of mutations. We will prove in the following that, given the appropriate T_B and $\mathcal{U}$, the adaptively encoded algorithm (Algorithm 2) is equivalent to the scaling adaptive Algorithm 3, where mutations are scaled according to a diagonal matrix σ.

Assumption 1. *Let `select` pick solutions only depending on the function values (not depending on the actual solutions), and let `reco` commute with $\boldsymbol{B}$, in that*

$$\boldsymbol{B} \cdot \mathtt{reco}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu) = \mathtt{reco}(\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu) \tag{6}$$

for all $\boldsymbol{B}$ and all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu$.

Algorithm 1: Baseline (μ, λ)-EA
The $R_i \in \mathbb{R}^n$ denote random vectors

```
let μ, λ positive integers and μ < λ
initialize {x_1, ..., x_μ} ∈ ℝ^n (a set of candidate solutions)
repeat
    x_i = reco(x_1, ..., x_μ) + R_i ,    for i = 1, ..., λ
    f_i = f(x_i) ,    for i = 1, ..., λ
    {x_1, ..., x_μ} ← select({x_i, f_i}_{i=1,...,λ})
until stopping criterion is met
```

Theorem 1. *Given Assumption 1, the adaptively encoded (μ, λ)-EA (Algorithm 2) implements the scaling adaptive (μ, λ)-EA (Algorithm 3), if*

$$\mathcal{U}(\sigma) = \texttt{adapt}_\sigma(\sigma) \tag{7}$$

holds with probability one.

Proof. Let initialize $\boldsymbol{B} = \sigma$ in Algorithm 2 and let the initial $\boldsymbol{x}_i$ be identical in both algorithms. We consider one iteration step of both algorithms.

First, we show that the same solutions are evaluated in both algorithms and consequently $f_i' = f_i$. That means, we have to prove that $\boldsymbol{B}\boldsymbol{x}_i'$ in line 7 of

Algorithm 2: Adaptively Encoded (μ, λ)-EA
Shaded areas implement the adaptive encoding. The begin-end block implements the baseline (μ, λ)-EA minimizing $f \circ \boldsymbol{B}$.

1 initialize $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \in \mathbb{R}^n$ (a set of solutions)
2 initialize $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ (a transformation matrix)
3 **repeat**
4 $\quad \boldsymbol{x}_i' = \boldsymbol{B}^{-1}\boldsymbol{x}_i\,, \quad$ for $i = 1, \ldots, \mu$ `// decode state`
5 $\quad$ **begin**
6 $\quad\quad \boldsymbol{x}_i' = \texttt{reco}(\boldsymbol{x}_1', \ldots, \boldsymbol{x}_\mu') + R_i\,, \quad$ for $i = 1, \ldots, \lambda$
7 $\quad\quad f_i' = f \circ \boldsymbol{B}(\boldsymbol{x}_i') = f(\boldsymbol{B}\boldsymbol{x}_i')$, for $i = 1, \ldots, \lambda$ `// encode to eval`
8 $\quad\quad \{\boldsymbol{x}_1', \ldots, \boldsymbol{x}_\mu'\} \leftarrow \texttt{select}(\{\boldsymbol{x}_i', f_i'\}_{i=1,\ldots,\lambda})$
9 $\quad$ **end**
10 $\quad \boldsymbol{x}_i \leftarrow \boldsymbol{B}\boldsymbol{x}_i'\,, \quad$ for $i = 1, \ldots, \mu$ `// encode`
11 $\quad \boldsymbol{B} \leftarrow \mathcal{U}(\boldsymbol{B})$ `// adapt encoding`
12 **until** stopping criterion is met

Algorithm 3: Scaling Adaptive (μ, λ)-EA
The shaded areas add to the baseline (μ, λ)-EA

1 initialize $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \in \mathbb{R}^n$ (a set of solutions)
2 initialize diagonal matrix σ (a scaling)
3 **repeat**
4 $\quad \boldsymbol{x}_i = \texttt{reco}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu) + \sigma \times R_i\,, \quad$ for $i = 1, \ldots, \lambda$
5 $\quad f_i = f(\boldsymbol{x}_i)\,, \quad$ for $i = 1, \ldots, \lambda$
6 $\quad \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \leftarrow \texttt{select}(\{\boldsymbol{x}_i, f_i\}_{i=1,\ldots,\lambda})$
7 $\quad \sigma \leftarrow \texttt{adapt}_\sigma(\sigma)$
8 **until** stopping criterion is met

Algorithm 2 equals to $\boldsymbol{x}_i$ in line 5 of Algorithm 3, for all $i = 1, \ldots, \lambda$.

$$\begin{aligned}
\boldsymbol{B}\boldsymbol{x}_i' &= \boldsymbol{B}\left(\texttt{reco}(\boldsymbol{x}_1', \ldots, \boldsymbol{x}_\mu') + R_i\right) \\
&= \boldsymbol{B}\,\texttt{reco}(\boldsymbol{B}^{-1}\boldsymbol{x}_1, \ldots, \boldsymbol{B}^{-1}\boldsymbol{x}_\mu) + \boldsymbol{B}R_i \\
&= \texttt{reco}(\boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{x}_\mu) + \sigma \times R_i \\
&= \texttt{reco}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu) + \sigma \times R_i \\
&= \boldsymbol{x}_i \qquad \text{for } i = 1, \ldots, \lambda
\end{aligned} \tag{8}$$

Second, we prove that all $\boldsymbol{x}_i$ are identical after one iteration step in both algorithms. According to Eq. (8), $\boldsymbol{B}\boldsymbol{x}_i'$ in line 7 of Algorithm 2 equals to $\boldsymbol{x}_i$ and therefore $f_i' = f_i$. Because `select` only depends on the function values, the individuals with the same indices are selected in both algorithms, respectively. Decoding in line 10 of Algorithm 2 assigns the desired equality.

Third, we remark that Eq. (7) implies that $\boldsymbol{B} = \sigma$ remains valid after the iteration step. □

We have proven that under Assumption 1, adaptive encoding can equivalently replaced adaptively scaled mutations. We conclude by a few remarks on the applicability of this result.

Remark 4. *The update, $\mathcal{U}$, can always be chosen, such that Eq.* (7) *holds, if all necessary information is included in the algorithms state.*

Remark 5. *Some typical recombination operators [1] commute with linear transformations.*

Remark 6. *The diagonal matrix σ in the scaling adaptive algorithm resembles (overall) step-size adaptation, if all diagonal entries remain identical.*

4.2 EDA (Estimation of Distribution Algorithm)

Algorithm 4 implements a very simple Estimation of Distribution Algorithm (EDA) [9], denoted as sEDA in the following. Only the mean vector of the distribution is estimated for the next generation. Adaptive encoding for the sEDA is elementary and outlined in Algorithm 5. Only the mean vector $\boldsymbol{m}$ needs to be transformed according to $T_B^{-1}(\boldsymbol{m}) = \boldsymbol{B}^{-1}\boldsymbol{m}$. The adaptation of the encoding is given in line 11 of Algorithm 5. The adaptation resembles the estimation of a covariance matrix in a common EDA. The common EDA is given in Algorithm 6, where the empirical covariance matrix is computed in line 7.

Algorithm 4: Simple EDA (sEDA)
$\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ denotes a normal distribution with covariance matrix $\boldsymbol{I}$; $i(f)$ indicates the index of the i-th best solution

1 initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean)
2 **repeat**
3 $\quad \boldsymbol{x}_i = \boldsymbol{m} + \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}), \quad$ for $i = 1, \ldots, \lambda$
4 $\quad f_i = f(\boldsymbol{x}_i), \quad$ for $i = 1, \ldots, \lambda$
5 $\quad \boldsymbol{m} \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}_{i(f)}$
6 **until** stopping criterion is met

The next theorem states that the adaptively encoded sEDA unconditionally implements the common EDA.

Theorem 2. *The adaptively encoded sEDA (Algorithm 5) implements the common EDA (Algorithm 6).*

Proof. We assume that initially $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{C}$ and that the initial $\boldsymbol{m}$ is the same in both algorithms. We prove that the same holds true after one iteration step.

Algorithm 5: Adaptively Encoded sEDA
Shaded lines implement the adaptive encoding; the begin-end block embraces the sEDA; $i(f)'$ indicates the index of the i-th best solution according to f_i'

initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean)
initialize $\boldsymbol{B} = I$
repeat
  $\boldsymbol{m}' = \boldsymbol{B}^{-1}\boldsymbol{m}$
  begin
    $\boldsymbol{x}_i' = \boldsymbol{m}' + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\,, \quad \text{for } i = 1, \ldots, \lambda$
    $f_i' = f(\boldsymbol{B}\,\boldsymbol{x}_i')\,, \quad \text{for } i = 1, \ldots, \lambda$
    $\boldsymbol{m}' \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}'_{i(f)'}$
  end
  $\boldsymbol{m} \leftarrow \boldsymbol{B}\boldsymbol{m}'$
  $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu}(\boldsymbol{B}\boldsymbol{x}'_{i(f)'} - \boldsymbol{m})(\boldsymbol{B}\boldsymbol{x}'_{i(f)'} - \boldsymbol{m})^{\mathrm{T}}$
until stopping criterion is met

Algorithm 6: Common EDA
Shaded areas add to the simple EDA

initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean)
initialize $\boldsymbol{C} = \boldsymbol{I}$ (empirical covariance matrix)
repeat
  $\boldsymbol{x}_i = \boldsymbol{m} + \mathcal{N}_i\left(\boldsymbol{0}, \boldsymbol{C}\right), \quad \text{for } i = 1, \ldots, \lambda$
  $f_i = f(\boldsymbol{x}_i)\,, \quad \text{for } i = 1, \ldots, \lambda$
  $\boldsymbol{m} \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}_{i(f)}$
  $\boldsymbol{C} \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu}(\boldsymbol{x}_{i(f)} - \boldsymbol{m})(\boldsymbol{x}_{i(f)} - \boldsymbol{m})^{\mathrm{T}}$
until stopping criterion is met

First, we recognize that the same solutions are evaluated in both algorithms, because

$$\begin{aligned}
\boldsymbol{B}\boldsymbol{x}_i' &= \boldsymbol{B}(\boldsymbol{m}' + \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I})) \qquad \text{in line 7 in Alg. 5} \\
&= \boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{m} + \boldsymbol{B}\mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}) \\
&= \boldsymbol{m} + \mathcal{N}_i\left(\boldsymbol{0}, \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\right) \\
&= \boldsymbol{m} + \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{C}) \\
&= \boldsymbol{x}_i
\end{aligned} \tag{9}$$

The equation holds almost surely, if the random number realizations are chosen correspondingly in both algorithms. Consequently, $f_i' = f_i$ and $i(f) = i(f)'$ for all $i = 1, \ldots, \lambda$. Therefore, the same solutions are used in the remaining operations and we find

$$\boldsymbol{m}_{\text{line 10}}^{\text{Alg. 5}} = \boldsymbol{B}\boldsymbol{m}' = \boldsymbol{B}\frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{x}_{i(f)'}' = \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{B}\boldsymbol{x}_{i(f)'}' = \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{x}_{i(f)} = \boldsymbol{m}_{\text{line 6}}^{\text{Alg. 6}} \ . \tag{10}$$

It remains to be shown that after the iteration step $\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{C}$. This follows directly from the choice of the update rule.

$$\begin{aligned}
\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} &\leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu}(\boldsymbol{B}\boldsymbol{x}_{i(f)'}' - \boldsymbol{m})(\boldsymbol{B}\boldsymbol{x}_{i(f)'}' - \boldsymbol{m})^{\mathrm{T}} \\
&= \frac{1}{\mu}\sum_{i=1}^{\mu}(\boldsymbol{x}_{i(f)} - \boldsymbol{m})(\boldsymbol{x}_{i(f)} - \boldsymbol{m})^{\mathrm{T}} \\
&= \boldsymbol{C}
\end{aligned} \tag{11}$$

□

5 A Universal Update Rule: $\mathrm{AE}_{\mathrm{CMA}}$

In order to define an adaptive encoding, we needed to specify the encoding of the algorithms state space, $T_B : S \to S$, and the update rule, $\mathcal{U}$, of the encoding matrix $\boldsymbol{B}$. In this section, our aim is to derive a universal update $\mathcal{U}$, leaving only the choice of T_B as remaining, algorithm specific design issue.

Previously, we have used two update rules, both in order to recover a known evolutionary algorithm. In the first case, we had chosen $\mathcal{U} = \texttt{adapt}_\sigma$ without further specification. In the second case, we had computed the decomposition of an empirical covariance matrix to derive $\boldsymbol{B}$. The latter depends in particular on a sufficiently large number of solutions μ. In the following, we explicate a more universal update rule for the representation matrix $\boldsymbol{B}$. The update is derived from the equations for the covariance matrix update in the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES [6, 8], denoted as $\mathrm{AE}_{\mathrm{CMA}}$, and explicated in Algorithm 7 $\mathrm{AE}_{\mathrm{CMA}}$-Update.

All parameters and coefficients of $\mathrm{AE}_{\mathrm{CMA}}$-Update are in detail discussed in Section 5.1. The state variables are $\boldsymbol{m}$, $\boldsymbol{p}$ and $\boldsymbol{C}$. The mean $\boldsymbol{m}$ is initialized to the mean of initial solutions of the search algorithm to which $\mathrm{AE}_{\mathrm{CMA}}$-Update is applied, and initially $\boldsymbol{p} = \boldsymbol{0}$ and $\boldsymbol{C} = \boldsymbol{I}$. The covariance matrix update is based on differences between new solutions and the (former) mean.

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 $i = 1, \ldots, \mu$. Then, the algorithm AE_{CMA}-Update implements the update equations for the evolution path, $\boldsymbol{p}$, and the covariance matrix, $\boldsymbol{C} = \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}$, in the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES.*

Proof. Assuming that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu$ are the μ best solutions in the recent iteration step, line 5 computes $\boldsymbol{m}$ according to Eq.(3) in [6]. Lines 7 and 10 of Algorithm 7 $\mathrm{AE_{CMA}}$-Update replicate the covariance matrix update equations (17) and (22) in [6] with added or renamed normalization coefficients, denoted as α. Substituting the coefficients as given results in the original equations. □

The Algorithm 7 $\mathrm{AE_{CMA}}$-Update implements the covariance matrix update of CMA-ES with additional coefficients α to be specified (see Section 5.1). In the context of CMA-ES, this update works well for any choice of μ [6].

Depending on the application of $\mathrm{AE_{CMA}}$-Update, a slow change of $\boldsymbol{B}$ might be desirable (see also Section 5.3). 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 are sorted in $\boldsymbol{D}$. We conjecture, that lines 11 to 13 can be replace by a cholesky decomposition, or, more promising, by an incremental cholesky update that ensures small changes, as long as c_1 and c_μ are small. In this case, it might be sufficient to only encode the solutions for the function evaluation and, as an approximation, completely abandon the encoding-decoding of the algorithms state.

In the next section, we will discuss the choice of the coefficients and of the remaining parameters of the procedure. Section 5.2 shows how $\mathrm{AE_{CMA}}$-Update recovers CMA-ES.

Algorithm 7: $\mathrm{AE_{CMA}}$-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 $w_i, c_{\mathrm{p}}, c_1, c_\mu$ from Section 5.1
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
4 $\boldsymbol{m}^- = \boldsymbol{m}$
5 $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_i$ `// Eq.(3) in [6]`
6 set scalars $\alpha_i \geq 0$, for $i = 0, \ldots, \mu$, cf. Sect. 5.1
7 $\boldsymbol{p} \leftarrow (1 - c_{\mathrm{p}})\, \boldsymbol{p} + \sqrt{c_{\mathrm{p}}\,(2 - c_{\mathrm{p}})}\, \alpha_0 (\boldsymbol{m} - \boldsymbol{m}^-)$ `// Eq.(17) in [6]`
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-`μ `matrix`
9 set scalar $\alpha_{\mathrm{p}} \geq 0$, cf. Sect. 5.1
10 $\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu)\, \boldsymbol{C} + c_1 \alpha_{\mathrm{p}}\, \boldsymbol{p}\boldsymbol{p}^{\mathrm{T}} + c_\mu \boldsymbol{C}_\mu$ `// Eq.(22) in [6]`
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`

5.1 Choice of Parameters

In Algorithm 7, the scalars α_{p} and α_i for $i = 0, \ldots, \mu$, need to be chosen. They normalize the input entries for the covariance matrix update (most of them are the difference between a new solution and the former mean). In the original CMA, we can derive the expected lengths of the input entries from the sampling procedure. Under random selection the normalized entries are distributed according to

$$\mathcal{N}(\mathbf{0}, \boldsymbol{C}) \quad = \quad \boldsymbol{B}\,\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \quad . \tag{12}$$

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

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

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

$$\alpha_i = \frac{\sqrt{n}}{\max\left(\frac{l_i}{\beta}, \operatorname*{median}_{j=1,\ldots,\mu}(l_j)\right)} \qquad \text{for } i = 1, \ldots, \mu \ , \tag{13}$$

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

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

Finally, we give the default settings for the constants used in $\mathrm{AE}_{\mathrm{CMA}}$-Update and discuss the choices in turn.

$c_{\mathbf{p}} = \frac{1}{\sqrt{n}}$ is the learning constant for the evolution path, which should be usually between $\frac{1}{\sqrt{n}}$ and $\frac{2}{n+1}$ [8]. For larger c_{p}, the effect of the evolution path will attenuate. The backward time horizon for the evolution path is roughly ${c_{\mathrm{p}}}^{-1}$. We choose as default the "conservative" limit of the useful range, *i.e.* a comparatively large c_{p}. The most conservative choice would be $c_{\mathrm{p}} = 1$.

[1] We ignore the case of denominators being zero, where the respective coefficient α can be set to any positive number.

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

$c_1 = \alpha_{\mathbf{c}} \frac{0.2}{(n+1.3)^2+\mu_{\mathbf{w}}}$ is the learning rate for the rank-one update in line 10 of $\mathrm{AE_{CMA}}$-Update (middle summand), with $\alpha_{\mathrm{c}} = 1$ as default. The denominator being quadratic in n reflects the degrees of freedom in the encoding matrix $\boldsymbol{B}$. The formula is derived as a simplification from the original formulation in [6].

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

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

The final parameter setting needs to be decided specifically for a given algorithm. We believe that the given guidelines will be usually sufficient to find good settings with reasonable effort. For $\mathrm{AE_{CMA}}$-CSA-ES (meaning CMA-ES), a good setting works across many objective functions and the identification needed to be conducted only once on a few simple test functions.

5.2 $\mathrm{AE_{CMA}}$ Recovers CMA-ES

The CMA-ES implements two separate adaptation mechanisms, one for step-size control and another for covariance matrix adaptation. Our next application of $\mathrm{AE_{CMA}}$ is based on cumulative step-size adaptation (CSA, also denoted as path length control). The $(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES is explicated in Algorithm 8. Its default parameters are

$$c_\sigma = \frac{\mu_{\mathrm{w}} + 2}{n + \mu_{\mathrm{w}} + 3} \quad \text{and} \quad d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\mathrm{w}} - 1}{n+1}} - 1\right) + c_\sigma \ . \tag{14}$$

For applying adaptive encoding, we choose the following encoding for the state variables in CSA-ES,

$$T_B : (\boldsymbol{m}, \boldsymbol{p}_\sigma, \sigma) \mapsto (\boldsymbol{B}\boldsymbol{m}, \boldsymbol{B}^\circ \boldsymbol{p}_\sigma, \sigma) \ . \tag{15}$$

The $\mathrm{AE_{CMA}}$-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES applies adaptive encoding according to Algorithm 7, $\mathrm{AE_{CMA}}$-Update, to the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES and is given in Algorithm 9. The μ best (encoded) solutions are used as input to $\mathrm{AE_{CMA}}$-Update (line 18 in Algorithm 9). The encoding T_B solely depends on the matrix $\boldsymbol{B}$, as $\boldsymbol{B}^\circ$ can be computed from $\boldsymbol{B}$ by normalizing its columns to length one.

Theorem 3 (Recovery of CMA-ES)**.** *Given T_B as in Eq.* (15) *and the scalars for Algorithm 7 $\mathrm{AE_{CMA}}$-Update in each iteration as given in Proposition 2, then the $\mathrm{AE_{CMA}}$-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES (Algorithm 9) implements the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES.*

Proof. As in our previous proofs, we assume the same initial state for $\mathrm{AE}_{\mathrm{CMA}}$-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES and $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES and first consider the sampled and

Algorithm 8: CSA-ES

initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean)
initialize $\boldsymbol{p}_\sigma = \boldsymbol{0}$ (evolution path)
initialize $\sigma > 0$ (step-size)
repeat
  $\boldsymbol{x}_i = \boldsymbol{m} + \sigma \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}), \quad \text{for } i = 1, \ldots, \lambda$
  $f_i = f(\boldsymbol{x}_i), \quad \text{for } i = 1, \ldots, \lambda$
  $\boldsymbol{m}^- = \boldsymbol{m}$
  $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}_{i(f)}$
  $\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\, \boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2 - c_\sigma)\mu_{\mathrm{w}}}\, \frac{1}{\sigma}(\boldsymbol{m} - \boldsymbol{m}^-)$
  $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1\right)\right)$
until stopping criterion is met

Algorithm 9: $\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES
Shaded lines implement the adaptive encoding, $\mathrm{AE}_{\mathrm{CMA}}$, including the update of $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$. The begin-end block marks the original CSA-ES

initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean)
initialize $\boldsymbol{p}_\sigma = \boldsymbol{0}$ (evolution path)
initialize $\sigma > 0$ (step-size)
initialize $\boldsymbol{B} = \boldsymbol{B}^\circ = \boldsymbol{I}$
repeat
  $\boldsymbol{m}' = \boldsymbol{B}^{-1}\boldsymbol{m}$
  $\boldsymbol{p}'_\sigma = \boldsymbol{B}^{\circ\mathrm{T}}\boldsymbol{p}_\sigma$
  begin
    $\boldsymbol{x}'_i = \boldsymbol{m}' + \sigma \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}), \quad \text{for } i = 1, \ldots, \lambda$
    $f'_i = f(\boldsymbol{B}\boldsymbol{x}'_i), \quad \text{for } i = 1, \ldots, \lambda$
    $\boldsymbol{m}'^- = \boldsymbol{m}'$
    $\boldsymbol{m}' \leftarrow \sum_{i=1}^{\mu} w_i \, \boldsymbol{x}'_{i(f)'}$
    $\boldsymbol{p}'_\sigma \leftarrow (1 - c_\sigma)\, \boldsymbol{p}'_\sigma + \sqrt{c_\sigma\,(2 - c_\sigma)\mu_{\mathrm{w}}}\, \frac{1}{\sigma}(\boldsymbol{m}' - \boldsymbol{m}'^-)$
    $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1\right)\right)$
  end
  $\boldsymbol{m} = \boldsymbol{B}\boldsymbol{m}'$
  $\boldsymbol{p}_\sigma = \boldsymbol{B}^\circ \boldsymbol{p}'_\sigma$
  $\mathrm{AE}_{\mathrm{CMA}}\text{-Update}(\{\boldsymbol{B}\boldsymbol{x}'_1, \ldots, \boldsymbol{B}\boldsymbol{x}'_\mu\})$
until stopping criterion is met

evaluated solutions $\boldsymbol{B}\boldsymbol{x}_i'$, for $i = 1, \ldots, \lambda$, in Algorithm 7. We have

$$\begin{aligned}\boldsymbol{B}\boldsymbol{x}_i' &= \boldsymbol{B}(\boldsymbol{m}' + \sigma\mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I})) \\ &= \boldsymbol{B}\boldsymbol{m}' + \sigma\boldsymbol{B}\mathcal{N}_i(\boldsymbol{0}, \boldsymbol{I}) \\ &= \boldsymbol{m} + \sigma\mathcal{N}_i\left(\boldsymbol{0}, \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\right) \\ &= \mathcal{N}_i\left(\boldsymbol{m}, \sigma^2\boldsymbol{C}\right) \quad , \end{aligned} \tag{16}$$

which is the equation for generating new solutions in CMA-ES (Eq. (2) in [6]). Consequently, the same (encoded) solutions are generated (given respective random number realizations) and evaluated in $\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES and CMA-ES. Also $i(f) = i(f)'$ and the corresponding solutions are selected for the remaining updates.

We need to show that $\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES recovers the update of all five state variables of CMA-ES, $\boldsymbol{m}$, $\boldsymbol{p}_\sigma$, σ, $\boldsymbol{p}$ and $\boldsymbol{C}$. We treat each variable in turn.

The new mean in $\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES obeys

$$\begin{aligned}\boldsymbol{m}_{\text{line 16}}^{\text{Alg. 9}} = \boldsymbol{B}\boldsymbol{m}' = \boldsymbol{B}\sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i(f)'}' &= \sum_{i=1}^{\mu} w_i \boldsymbol{B}\boldsymbol{x}_{i(f)'}' \\ &= \sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i(f)} \quad , \end{aligned} \tag{17}$$

which is the equation for updating the mean in CMA-ES (Eq. (3) in [6]). Next, we investigate the evolution path for the step-size, starting from line 17.

$$\begin{aligned}\boldsymbol{p}_\sigma &= \boldsymbol{B}^\circ \boldsymbol{p}_\sigma' \\ &\leftarrow \boldsymbol{B}^\circ\left((1-c_\sigma)\,\boldsymbol{p}_\sigma' + \sqrt{c_\sigma\,(2-c_\sigma)\mu_{\mathrm{W}}}\,\frac{1}{\sigma}(\boldsymbol{m}' - \boldsymbol{m}'^{-})\right) \\ &= (1-c_\sigma)\,\boldsymbol{B}^\circ\boldsymbol{p}_\sigma' + \sqrt{c_\sigma\,(2-c_\sigma)\mu_{\mathrm{W}}}\,\frac{1}{\sigma}\boldsymbol{B}^\circ(\boldsymbol{m}' - \boldsymbol{m}'^{-}) \\ &= (1-c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2-c_\sigma)\mu_{\mathrm{W}}}\,\frac{1}{\sigma}\boldsymbol{B}^\circ\boldsymbol{B}^{-1}(\boldsymbol{m} - \boldsymbol{m}^{-}) \end{aligned} \tag{18}$$

We compute the rightmost term to

$$\begin{aligned}\boldsymbol{B}^\circ\boldsymbol{B}^{-1}(\boldsymbol{m} - \boldsymbol{m}^{-}) &= \boldsymbol{B}^\circ(\boldsymbol{B}^\circ\boldsymbol{D})^{-1}(\boldsymbol{m} - \boldsymbol{m}^{-}) \\ &= \boldsymbol{B}^\circ\boldsymbol{D}^{-1}\boldsymbol{B}^{\circ\mathrm{T}}(\boldsymbol{m} - \boldsymbol{m}^{-}) \quad , \end{aligned} \tag{19}$$

where $\boldsymbol{B} = \boldsymbol{B}^\circ\boldsymbol{D}$ holds according to Algorithm 7. Equations (18) and (19) recover the update rule for the evolution path $\boldsymbol{p}_\sigma$ in CMA-ES (Eq. (23) in [6]).

The update of step-size σ computes $\|\boldsymbol{p}_\sigma'\| = \|\boldsymbol{B}^\circ\boldsymbol{p}_\sigma\|$. Because $\boldsymbol{B}^\circ$ is orthogonal we have $\|\boldsymbol{B}^\circ\boldsymbol{p}_\sigma\| = \|\boldsymbol{p}_\sigma\|$. Consequently, the step-size update is identical in $\mathrm{AE}_{\mathrm{CMA}}$-CSA-ES and CMA-ES.

Finally, completing the proof, $\boldsymbol{p}$ and $\boldsymbol{C}$ are updated in Algorithm $\mathrm{AE}_{\mathrm{CMA}}$-Update according to the CMA-ES, see Proposition 2. □

Theorem 3 supports the hypothesis that $\mathrm{AE}_{\mathrm{CMA}}$-Update is an efficient way to update the representation matrix $\boldsymbol{B}$, as CMA-ES is known to efficiently adapt the principle axes of the coordinate system, where the independent sampling takes place. In the next section, another application of $\mathrm{AE}_{\mathrm{CMA}}$-Update is sketched.

5.3 Application of $\mathrm{AE}_{\mathrm{CMA}}$-Update

The scaling adaptive (μ, λ)-EA, Algorithm 3, samples new solutions without dependencies between variables in the given coordinate system, because σ is a diagonal matrix. Rendering the scaling adaptive (μ, λ)-EA coordinate system independent means that the distribution can reveal correlations with respect to the given coordinate system (if $\sigma \neq \boldsymbol{I}$ is not the identity).

For the scaling adaptive (μ, λ)-EA the encoding

$$T_B : (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu, \sigma, y) \mapsto (\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu, \sigma, y) \tag{20}$$

suggests itself, where y denotes all further state variables.

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. A transformation of the diagonal of σ with $\boldsymbol{B}$ can lead, by chance, to very small entries—a very undesirable result for a step-size. Because σ is not encoded, it is important that changes of $\boldsymbol{B}$ are modest.

A vector $\boldsymbol{r}$, that denotes a direction rather than a solution point in state space (as for example a velocity in Particle Swarm Optimization [2]), would arguably map to $\boldsymbol{B}\boldsymbol{r}$.

Using $\boldsymbol{B}^\circ$ instead of $\boldsymbol{B}$ in Eq. (20) is a possible alternative. Then, the step-size matrix σ needs to learn the scaling which can be otherwise provided by the diagonal matrix $\boldsymbol{D}$.

We experimented using the Cauchy distribution for the random vector R_i in Algorithm 3 as a rather different example compared to CSA. The adaptive encoding was successfully applied in this case and made the Baseline (μ, λ)-EA roughly thousand times faster on proto-typical non-separable problems.

6 Summary and Conclusions

We have outlined an adaptive change of representation in continuous domain search, denoted as adaptive encoding (AE). The idea is simple: after each iteration step, (i) the algorithms state is "encoded", (ii) the encoding is adapted, and (iii) the algorithms state is "decoded" again for the next iteration, using the updated inverse encoding. Further, candidate solutions are encoded for their evaluation on the objective function, respectively. The implications from this simple idea are surprisingly far-reaching.

1. Well-known evolutionary algorithms can be recovered from basic algorithms in that a specific adaptive encoding is applied.

2. The update of the covariance matrix in the CMA-ES can be entirely formulated by adaptive encoding, as adaptation of a representation matrix $\boldsymbol{B}$ (Proposition 2). The respective update rule is given in $\mathrm{AE}_{\mathrm{CMA}}$-Update (Algorithm 7) and can recover the CMA-ES (Theorem 3).

3. Adaptive encoding can be applied to any search algorithm. Consequently, using the $\mathrm{AE}_{\mathrm{CMA}}$-Update, the "covariance matrix update" of CMA becomes applicable to any continuous domain search algorithm. We anticipate successful applications in particular for population based, stochastic algorithms.

References

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. Natural Computing, 1(1):3–52, 2002.

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

[3] W. Davidon. Variable Metric Method for Minimization. SIAM Journal on Optimization, 1:1, 1991.

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

[5] N. Hansen. Invariance, self-adaptation and correlated mutations in evolution strategies. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, editors, Parallel Problem Solving from Nature—PPSN VI, Proceedings, pages 355–364, Paris, 2000. Springer, Berlin.

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

[7] N. Hansen, F. Gemperle, A. Auger, and P. Koumoutsakos. When do heavy-tail distributions help? Parallel Problem Solving from Nature-PPSN IX, 4193:62–71, 2006.

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

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

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399