A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity

Raymond Ros, Nikolaus Hansen

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity

Raymond Ros — Nikolaus Hansen

N° ????

April 2008

Thème COG

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

A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity

Raymond Ros, Nikolaus Hansen

Thème COG — Systèmes cognitifs
Équipe-Projet TAO

Rapport de recherche n° ???? — April 2008 — 22 pages

Abstract: This report proposes a simple modification of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for high dimensional objective functions that reduces the internal time and space complexity from quadratic to linear. The covariance matrix is constrained to be diagonal and the resulting algorithm, sep-CMA-ES, samples each coordinate independently. Because the model complexity is reduced, the learning rate for the covariance matrix can be increased. Consequently, on essentially separable functions, sep-CMA-ES significantly outperforms CMA-ES. For dimension larger than 100, even on the non-separable Rosenbrock function, the sep-CMA-ES needs fewer function evaluations than CMA-ES.

Key-words: Evolution Strategy, performance assessment, separability, Covariance Matrix Adaptation, time complexity

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

Une Simple Modification dans CMA-ES pour une Complexité Spatiale et Temporelle Linéaire

Résumé : Pas de résumé

Mots-clés : Pas de motclef

1 Introduction

The search space dimensionality, n, plays an essential role in real parameter $\mathbb{R}^n$ optimisation, where a non-linear *objective function*, $f : \mathbb{R}^n \to \mathbb{R}$, is to be minimised. Its importance is emphasised by the notion of *curse of dimensionality*: the search space volume increases exponentially with n, making space filling sampling intractable even for moderate dimensionalities. The curse of dimensionality is only a concern if *dependencies* between parameters of the objective function are prevalent: when the parameters are independent, the search can be conducted along coordinate axes in *one-dimensional* subspaces altogether by n one-dimensional search procedures. Consequently, difficult real parameter optimisation problems exhibit essential dependencies between the parameters—and learning dependencies turns out to be decisive for solving these difficult problems. In evolutionary computation the issue of learning dependencies in real parameter search spaces is successfully addressed by covariance matrix adaptation (CMA) [4]. The CMA learns all pair-wise dependencies between all parameters by updating a covariance matrix for the sample distribution. The update mechanism is independent of the given coordinate system. The CMA was introduced for evolution strategies (ESs) but recently applied also in Evolutionary Gradient Search [1]. In learning *all* pair-wise dependencies, the CMA algorithm has an internal computational complexity of at least $\mathcal{O}(n^2)$. Empirical results indicate that, in order to learn the complete covariance matrix, the *number of objective function evaluations* usually scales sub-quadratically with n [3, 4].

In what follows, we will assume a black-box scenario in which function evaluations on f are the only way to gather insights into the nature of f (and therefore to make a reasonable proposal for a solution vector with small function value). The number of function evaluations is regarded as *search costs*. Furthermore, we call a function f *separable* if the parameters of f are independent, in that the global optimum can be obtained by n one-dimensional optimisation procedures along the coordinate axes for any given initial point.

1.1 Motivation

A principle limitation of CMA results from the number of so-called strategy parameters, $\frac{n^2+n}{2}$, that needs to be adapted in the covariance matrix. The number of strategy parameters, in other words the degrees of freedom in the covariance matrix, can dominate the search costs (number of objective function evaluations to reach a target function value). The full learning task scales roughly with n^2 (see *e.g.* [4]). Therefore, for large search space dimensionality, achieving a better scaling property might be attractive. A second limitation of CMA, probably less important, lies in its *internal* computational complexity and is explicated in the following.

- Sampling a general multivariate normally distributed random vector has a complexity of n^2 (per sampled n-dimensional vector). A matrix-vector multiplication needs to be conducted.
- Updating the covariance matrix has a complexity of $(\mu + 1)n^2$. The so-called rank-μ update [3] amounts to μ covariance matrix updates, one for each parent vector.

- Factorising the covariance matrix $\boldsymbol{C}$ into $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{C}$ has a complexity of n^3. The factorisation is needed to sample the multivariate normal distribution with covariance matrix $\boldsymbol{C}$. In the CMA-ES, usually an eigendecomposition is used to compute a *symmetric* (unique) factorisation matrix $\boldsymbol{A}$. A symmetric factorisation allows to compute the conjugate evolution path for step-size adaptation accurately, and it allows to track the eigenvalues of the covariance matrix, which often proves to be very useful in practice. Also the eigendecomposition has a complexity of n^3. Usually this computation is postponed until after $n/10$ generations and consequently slightly outdated distributions are sampled (with an insignificant effect on the performance) [4]. Consequently the complexity of this step becomes n^2 per generation.[1]

In summary, several steps in the CMA algorithm have a computational complexity of $\Theta\left(n^2\right)$.

The most obvious option toward achieving better scaling behaviour for the search costs is to *reduce the degrees of freedom* in the covariance matrix. We can think of several ways, and parameterisations, for reducing the degrees of freedom, resulting in a family of potentially useful modifications of CMA-ES, which *trade off model complexity for learning speed*. This trade will be a bad buy whenever the full model complexity is indispensable in order to efficiently solve the underlying problem. Otherwise search costs can be reduced according to a reduced learning period.

In this report we pursue, as a first step, the maybe simplest modification of CMA that reduces the degrees of freedom in the covariance matrix to n: we will devise a *small* modification of CMA, denoted as sep-CMA, that enables to learn the diagonal elements of the covariance matrix in linear time. The resulting sampled distribution being independent w.r.t. the coordinate system, it is therefore separable: the prefix 'sep' stands for separable. Even though we interpret this modification rather as a preliminary step, this step reveals some interesting perspectives even on its own.

- The sep-CMA-ES can serve as a **baseline comparison**. First, in comparison with CMA-ES, it can empirically measure profits and losses from learning *the dependencies* in the CMA framework. Experiments with sep-CMA-ES can in particular quantify the loss from having the ability to learn the complete covariance matrix in CMA, even if only a scaling of independent parameters needs to be acquired.

  Second, when CMA-ES is outperformed by another algorithm, $\mathcal{A}$, then sep-CMA-ES can give a positive answer to the question of whether the advantage is due to an exploitation of the (partial) separability of the function. Only if $\mathcal{A}$ also outperforms sep-CMA-ES, is a deeper explanation needed—which in this case could still be the exploitation of separability, but in a way sep-CMA-ES is not able to achieve.

  Third, it will serve as a baseline comparison for methods we plan to develop, that only learn a few dependencies with linear time and space complexity.

[1] More precisely, the computation is postponed until after ${c_{\text{cov}}}^{-1} n^{-1}/5$ generations, where the learning rate for the covariance matrix, c_{cov}, equals approximately $2\,n^{-2}$ for small populations. As the learning rate depends on the parent population size, the complexity becomes n^2 per parent vector.

In all cases, the implementation of a smallest possible modification is substantial.

- On non-trivial **large scale problems** (say $n \approx 1000$ or larger) with only moderate dependencies, sep-CMA-ES becomes a valuable alternative to CMA-ES, because it might perform considerably faster (in terms of number of function evaluations). On fast to evaluate objective functions, also a large advantage will be obtained when regarding overall CPU time. It is important to note that, sep-CMA-ES will also be able to exploit a *large population size*, just like CMA-ES [2, 3].

Objectives of this Report In this report we address two main objectives.

- Formulating a smallest possible modification of CMA, denoted as sep-CMA, that can learn a scaling of parameters in linear time, and, as an important part of algorithm design, *carefully re-identifying the learning rate for the covariance matrix.* The sep-CMA-ES is the first derandomised evolution strategy to our knowledge that will learn a scaling of variables in linear time *and* is feasible for large populations.

- Comparing the performance of sep-CMA-ES on both separable *and non-separable* functions to the performance of CMA-ES, as well as those of previously proposed evolution strategies with individual step-size adaptation and of other algorithms that exploit problem separability. In particular, sep-CMA-ES will turn out to be advantageous *not only* on separable functions but on non-separable function as well.

1.2 Favourably Scaling CMA Variants: Previous Works

The learning rate for the covariance matrix in CMA-ES is roughly proportional to n^{-2}, meaning the adaptation of the full covariance matrix needs roughly $\mathcal{O}(n^2)$ function evaluations. Nevertheless, a constant number of long axes of the distribution can be learnt in $\mathcal{O}(n)$, that is, in a linear number of function evaluations: the so-called *cumulation* allows for learning subspaces, in the space of covariance matrices, in linear time. Therefore the scale-up for the number of objective function evaluations is linear on the cigar function and on smooth ridge functions [3, 4].[2]

Some previous approaches reduces the overall time complexity of CMA-ES. A $\mathcal{O}\left(n^2\right)$ incremental Cholesky update [6] of the covariance matrix was proposed as a replacement for the eigendecomposition in CMA. The use of the Cholesky update is incompatible with the concept of evolution path, thus the step-size adaptation was replaced with a (1+1)-CMA-ES using an improved implementation of the 1/5-th success rule. The resulting CMA-ES variant is better than the original CMA-ES on some unimodal functions but is less effective for the class of functions for which the evolution path greatly improves performances.

Some ESs, which were introduced prior to CMA-ES, already implemented key features of the CMA-ES and scale favourably with the dimension. In [8],

[2]The cumulation parameter c_c must be chosen according to $c_\mathrm{c}^{-1} \propto n$ in order to observe the linear scaling behaviour.

a derandomised mutative step-size procedure was introduced to adapt individual step-sizes using cumulation and resulted in a (1, λ) ES with n strategy parameters. An extension of this derandomised step-size adaptation denoted AII-ES and introduced in [5] consists in combining it with the adaptation of one direction. This ES updates $2n$ strategy parameters.

The MVA-ES algorithm [9] uses the adaptation of the main (mutation) vector. This modification renders the time complexity of the algorithm to $\mathcal{O}(n)$ as the strategy parameters of the adaptation process are reduced to the length of the main vector, n. The MVA-ES algorithm is efficient in the specific case of objective functions having a single preferred mutation direction.

Filling the gap between CMA-ES and MVA-ES, L-CMA-ES [7] is a variant in which a parameter m allows to control the dimensionality of the representation of the mutation distribution. Their idea is to restrain the optimisation of the initial n-dimensional problem to that of its m main components. For the two extremes, if $m = 1$ L-CMA-ES is similar in substance to MVA-ES and if $m = n$ it is equivalent to the original CMA-ES.

In this report, as opposed to previous works, we address another subspace of strategy parameters that can be easily identified: the diagonal of the covariance matrix.

The **remainder of this report** is organised as follows. The Section 2 will present the CMA-ES algorithm and from this description, we will introduce sep-CMA-ES in Section 3. We explain how the problem of reducing the strategy parameters was addressed as well as compare the time complexity of sep-CMA-ES to that of CMA-ES. In Section 4, we propose to analyse sep-CMA-ES on some standard test functions. Results from these experiments are discussed in Section 5 and provide insights that we will develop in the last section of this report.

2 CMA-ES

The CMA-ES is a state-of-the-art continuous domain evolution strategy algorithm, introduced by [4] and described in [2, 3] that uses a covariance matrix adaptation combined with the computation of an evolution path. The CMA-ES benefits from larger population sizes by the use of a rank-μ update along with a weighted recombination of offspring. It is more precisely denoted as $(\mu/\mu_W, \lambda)$ CMA-ES.

Offspring for the generation $g + 1$ are sampled according to the following equation:

$$\boldsymbol{x}_k^{(g+1)} \sim \langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)} + \sigma^{(g)} \underbrace{\boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} \boldsymbol{z}_k^{(g+1)}}_{\mathcal{N}\left(0, \boldsymbol{C}^{(g)}\right)}, \; k = 1, \ldots, \lambda \tag{1}$$

where:

- λ is the population size, its default value being $4 + \lfloor 3 \ln(n) \rfloor$;
- μ is the number of the best individuals that will be recombined, the default value of μ is $\lfloor \frac{\lambda}{2} \rfloor$;

- $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(g)}$ is the weighted mean of the μ best individuals at generation g, $\boldsymbol{x}_{i:\lambda}^{(g)}$ denotes the i-th best out of the λ individuals ranked by function value;
- w_i, $i = 1, \ldots, \mu$ are the recombination weights, they are positive and sum to one. Setting the w_i to $1/\mu$ corresponds to an intermediate recombination. We use this expression: $w_i = \frac{\ln(\mu+1)-\ln(i)}{\sum_{j=1}^{\mu} \ln(\mu+1)-\ln(j)}$ that favours the best ranked individuals more;
- $\mathcal{N}(0, \boldsymbol{M})$ denotes independent realisations of the multi-variate normal distribution with covariance matrix $\boldsymbol{M}$;
- the random vectors $\boldsymbol{z}_k^{(g+1)}$ are $\mathcal{N}(0, \boldsymbol{I})$ distributed, and just as for the $\boldsymbol{x}_k^{(g)}$, we can compute their weighted mean: $\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}^{(g+1)}$;
- $\boldsymbol{B}^{(g)}$ is an orthogonal $n \times n$ matrix and $\boldsymbol{D}^{(g)}$ is a diagonal $n \times n$ matrix obtained from the eigendecomposition of $\boldsymbol{C}^{(g)}$, $\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\left(\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\right)^T = \boldsymbol{C}^{(g)}$. The covariance matrix, $\boldsymbol{C}^{(g)}$, is symmetric positive definite, its default initial value is $\boldsymbol{I}$;
- $\sigma^{(g)} \in \mathbb{R}^+$ is the step-size.

Both the global step-size $\sigma^{(g)}$ and the covariance matrix $\boldsymbol{C}^{(g)}$ are iteratively adapted. The path length control consists in adapting the global step size σ^g by an evolution path.

$$\boldsymbol{p_\sigma}^{(g+1)} = (1 - c_\sigma)\boldsymbol{p_\sigma}^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)} \underbrace{\sqrt{\mu_{\mathrm{eff}}}\boldsymbol{B}^{(g)}\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)}}_{\frac{\sqrt{\mu_{\mathrm{eff}}}}{\sigma^{(g)}}\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\left(\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g+1)} - \langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)}\right)} \tag{2}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\| \boldsymbol{p_\sigma}^{(g+1)} \|}{E(\| \mathcal{N}(0, \boldsymbol{I}) \|)} - 1\right)\right) \tag{3}$$

where:

- $c_\sigma \in [0, 1[$ is the time constant for the adaptation of the step size $\sigma^{(g+1)}$, its default value is: $\frac{\mu_{\mathrm{eff}}+2}{n+\mu_{\mathrm{eff}}+3}$;
- $\mu_{\mathrm{eff}} = (\sum_{i=1}^{\mu} w_i)^2 / \sum_{i=1}^{\mu} w_i^2$ denotes the 'variance-effective selection mass', μ_{eff} is equal to μ if $w_i = 1/\mu$;
- $d_\sigma > 0$ is a damping factor, default value is: $1 + 2\max\left(0, \sqrt{\frac{\mu_{\mathrm{eff}}-1}{n+1}} - 1\right) + c_\sigma$.

The initial value of the evolution path is $\boldsymbol{p_\sigma}^{(0)} = \boldsymbol{0}$. The initial step-size $\sigma^{(0)}$ is a problem-dependent parameter.

The adaptation of $\boldsymbol{C}^{(g)}$ is done by the evolution path $\boldsymbol{p}_{\mathrm{c}}{}^{(g+1)}$ and by the μ-weighted difference vectors between the recent parents and $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)}$:

$$\boldsymbol{p}_{\mathrm{c}}{}^{(g+1)} = (1 - c_{\mathrm{c}})\boldsymbol{p}_{\mathrm{c}}{}^{(g)} + H_\sigma^{(g+1)}\sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})} \underbrace{\sqrt{\mu_{\mathrm{eff}}}\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)}}_{\frac{\sqrt{\mu_{\mathrm{eff}}}}{\sigma^{(g)}}\left(\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g+1)} - \langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)}\right)} \tag{4}$$

$$\boldsymbol{C}^{(g+1)} = (1 - c_{\text{cov}})\boldsymbol{C}^{(g)} + \frac{1}{\mu_{\text{cov}}} c_{\text{cov}} \boldsymbol{p}_{\text{c}}{}^{(g+1)} \left(\boldsymbol{p}_{\text{c}}{}^{(g+1)}\right)^T + c_{\text{cov}} \left(1 - \frac{1}{\mu_{\text{cov}}}\right) \underbrace{\sum_{i=1}^{\mu} w_i \boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} \boldsymbol{z}_{i:\lambda}^{(g+1)} \left(\boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} \boldsymbol{z}_{i:\lambda}^{(g+1)}\right)^T}_{\sum_{i=1}^{\mu} \frac{w_i}{\sigma^{(g)2}} \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \langle \boldsymbol{x} \rangle_{\text{w}}^{g}\right) \left(\boldsymbol{x}_{i:\lambda}^{(g+1)} - \langle \boldsymbol{x} \rangle_{\text{w}}^{g}\right)^T} \quad (5)$$

where:

- $c_{\text{c}} \in [0, 1]$ has the same role as c_σ in Eq.(2), it is a time a constant for the adaptation of the covariance matrix, its default value is $\frac{4}{n+4}$;
- $H_\sigma^{(g+1)}$ equals to one if $\frac{\|\boldsymbol{p}_{\boldsymbol{\sigma}}{}^{(g+1)}\|}{\sqrt{1-(1-c_\sigma)^{2(g+1)}}} < (1.4 + \frac{2}{n+1}) E(\| \mathcal{N}(0, \boldsymbol{I}) \|)$, and zero otherwise. The condition on H_σ is slightly different from [2], which can be noticeable in case of large dimension and small λ, say 9, together with either too a small initial step-size σ or on time-variant objective functions.
- $\frac{1}{\mu_{\text{cov}}} \in [0, 1]$ is a coefficient that controls the emphasis on the evolution path term, μ_{cov} default value is μ_{eff};
- $c_{\text{cov}} \in [0, 1]$ is the learning rate, its default value is:

$$c_{\text{cov}\,\text{default}} = \frac{1}{\mu_{\text{cov}}} \frac{2}{\left(n + \sqrt{2}\right)^2} + \left(1 - \frac{1}{\mu_{\text{cov}}}\right) \min\left(1, \frac{2\mu_{\text{cov}} - 1}{(n+2)^2 + \mu_{\text{cov}}}\right) \quad (6)$$

- $\boldsymbol{z}_{i:\lambda}^{(g+1)}$ denotes the i-th best out of the λ individuals ranked by function value.

The initial values are: $\boldsymbol{p}_{\text{c}}{}^{(0)} = \boldsymbol{0}$, $\boldsymbol{C}^{(0)} = \boldsymbol{I}$. The parameter $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)}$ is problem-dependent.

The whole process of sampling new individuals (Eq. 1) and updating the internal strategy parameters (covariance matrix) (Eq. 2,3,4,5) is iterated until a stopping criterion is reached.

3 sep-CMA-ES

In this section we introduce a CMA-ES variant, denoted as sep-CMA-ES, designed to reduce the time complexity of the adaptation of the covariance matrix $\boldsymbol{C}^{(g)}$. Compared to the default CMA-ES, two simple changes are undertaken. (i) The covariance matrix $\boldsymbol{C}$ is constrained to be diagonal, (ii) the learning rate c_{cov} is increased. This results in having an ES that samples independently w.r.t. the coordinate system using n individual variances that are updated.

3.1 Description of sep-CMA-ES

For obtaining sep-CMA-ES from the CMA-ES algorithm only the update of the covariance matrix in Eq. (5) is modified: the covariance matrix $\boldsymbol{C}^{(g)}$ remains

a diagonal matrix, because its update is restrained to the diagonal elements of the matrix, and Eq. (5) becomes:

$$c_{jj}^{(g+1)} = (1 - c_{\text{cov}})c_{jj}^{(g)} + \frac{1}{\mu_{\text{cov}}} c_{\text{cov}} \left(p_c^{(g+1)}\right)_j^2 + c_{\text{cov}} \left(1 - \frac{1}{\mu_{\text{cov}}}\right) \sum_{i=1}^{\mu} w_i \, c_{jj}^{(g)} \left(z_{i:\lambda}^{(g+1)}\right)_j^2, \quad j = 1, \ldots, n \quad (5')$$

where, for $j = 1, \ldots, n$, the c_{jj} are the diagonal elements of $\boldsymbol{C}^{(g)}$ and the $\left(z_{i:\lambda}^{(g+1)}\right)_j$ are the j-th component of $\boldsymbol{z}_{i:\lambda}^{(g+1)}$.

Whereas in the CMA-ES an eigendecomposition was needed before Eq. 1, now this step is reduced to taking the square root of the diagonal elements of $\boldsymbol{C}^{(g)}$ which has $\mathcal{O}(n)$ time complexity. The complexity of all other equations involving matrix operations is reduced since they now involve vector operations. The complexity reduction is made possible by the reduction of the model complexity but in the process it loses what made CMA-ES able to learn the dependencies of the parameters of the objective function. In other words, sep-CMA-ES loses the property of being rotationally invariant that CMA-ES has.

3.2 Identification of c_{cov}

While in the default CMA-ES algorithm, $\boldsymbol{C}^{(g)}$ has $n + \frac{n^2-n}{2}$ degrees of freedom, in the sep-CMA-ES algorithm $\boldsymbol{C}^{(g)}$ only has n. Thus, the default value of the learning rate c_{cov} appearing in Eq. (5') should be readjusted. The behaviour of sep-CMA-ES is studied using different values for the learning rate. We use two measures to qualify the effects of the varying learning rate on sep-CMA-ES: (i) the ratio of the largest and smallest eigenvalue of the final covariance matrix $\boldsymbol{C}$ (denoted in the following as *final axis ratio*) which should be close to the square root of the condition number of the objective function, (ii) the number of function evaluations to reach a given target value.

The unimodal and separable sphere function, $f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$ was used as test function. We also tested the algorithm on the ellipsoid function,

$$f_{\text{elli}}^{\beta}(\boldsymbol{x}) = \sum_{i=1}^{n} \beta^{\frac{i-1}{n-1}} y_i^2$$

where the parameter β is the condition number (ratio of the longest and smallest axis lengths) and $\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$. The coordinate system $\boldsymbol{Q}$ is either equal to $\boldsymbol{I}$ for the axis-parallel function or an orthogonal $n \times n$ matrix with each column vector $\boldsymbol{q_j}$ $(j = 1, \ldots, n)$ being a uniformly distributed unit vector for the rotated function. The ellipsoid function is non-separable in the general case, convex quadratic and unimodal.

3.2.1 Experimental Set-up for the Identification of c_{cov}

The implementation of the sep-CMA-ES algorithm that we use in our experiments is derived from a CMA-ES implementation in Scilab. We set the learning rate c_{cov} to be the product of $c_{\text{cov default}}$ (defined in Sect. 2) by a factor ranging

from 0 to 320. Other than c_{cov}, the parameters of the algorithms are set to the default values given in Sect. 2 for both the CMA-ES and sep-CMA-ES.

While the default initial value of the covariance matrix is $\boldsymbol{C}^{(0)} = \boldsymbol{I}$, both algorithms are also tested on the sphere function, f_{sphere}, with $\boldsymbol{C}^{(0)}$ set to a $n \times n$ diagonal matrix which diagonal elements are $(1, \beta^{\frac{1}{n-1}}, \beta^{\frac{2}{n-1}}, \ldots, \beta)$ (with $\beta = 10^6$).

The dimension n of the problem is chosen in the interval $[2, 80]$. The initial step-size is $\sigma^{(0)} = 1$ and the initial distribution is centred on $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = (5, \ldots, 5)^T$. Runs fail if, before the target function value is attained, either the axis ratio is larger than 10^8 (which could lead to some precision issues in the eigendecomposition of the covariance matrix) or the maximum number of evaluations is reached. For each experimental set-up, 21 runs are done except for $c_{\text{cov}} = 0$ (no learning occurs) for which 5 runs are done. As for the coordinate system, $\boldsymbol{Q}$, we use 21 different rotation matrices.

3.2.2 Discussion on the Learning Rate in sep-CMA-ES

For the larger values of c_{cov}, sep-CMA-ES becomes less reliable since no run succeeds: they are stopped to prevent having precision issues due to the handling of covariance matrices with axis ratios as large as 10^{16}. Results in Fig. 1 on the sphere function show the final axis ratios raise in conjunction with lower performances (*i.e.* more function evaluations) of the algorithm as the learning rate increases.

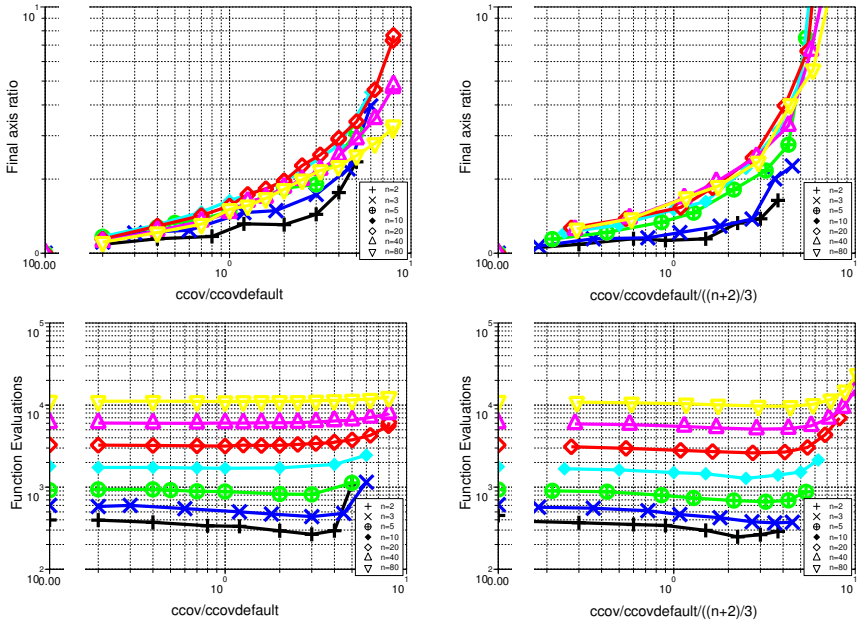
On the sphere function, when the condition number of the covariance matrix is initialised to β instead of 1, increasing the learning rate improves the performance until it degrades because the learning rate is too high.

Choosing the right value for learning rate means to make sure that the adaptation process is successful and that the performance of the algorithm is reliable. This is observable for intermediate values of c_{cov} where the final axis ratio is close the square root of the condition number of the objective function which is 1 in the case of the sphere function. In the case of the ill-conditioned and non-separable ellipsoid function, $f_{\text{elli}}^{10^3}$, choosing a learning rate for which the final axis ratios are close to $\sqrt{10^3} \approx 32$ yields performances that are reliable.

Results of the CMA-ES for which the default value $c_{\text{cov}\,\text{default}}$ is adjusted show the boundary values of the ratio $\frac{c_{\text{cov}}}{c_{\text{cov}\,\text{default}}}$ to be pretty close whatever the dimension considered (left subfigures in Fig. 1 and Fig. 2). Out of this domain, unreliable behaviours are to be expected. To have a similar domain as the dimension varies for the sep-CMA-ES algorithm, the ratio $\frac{c_{\text{cov}}}{c_{\text{cov}\,\text{default}}}$ had to be divided by $\frac{n+2}{3}$. In all the following experiments, it is this default value, $c_{\text{cov}} = \frac{n+2}{3} c_{\text{cov}\,\text{default}}$, that will be used for sep-CMA-ES.

4 Test Functions and Methods

In this section we describe the functions and methods used to test the sep-CMA-ES algorithm. We will compare the performances of sep-CMA-ES to those of CMA-ES.



Default CMA-ES sep-CMA-ES

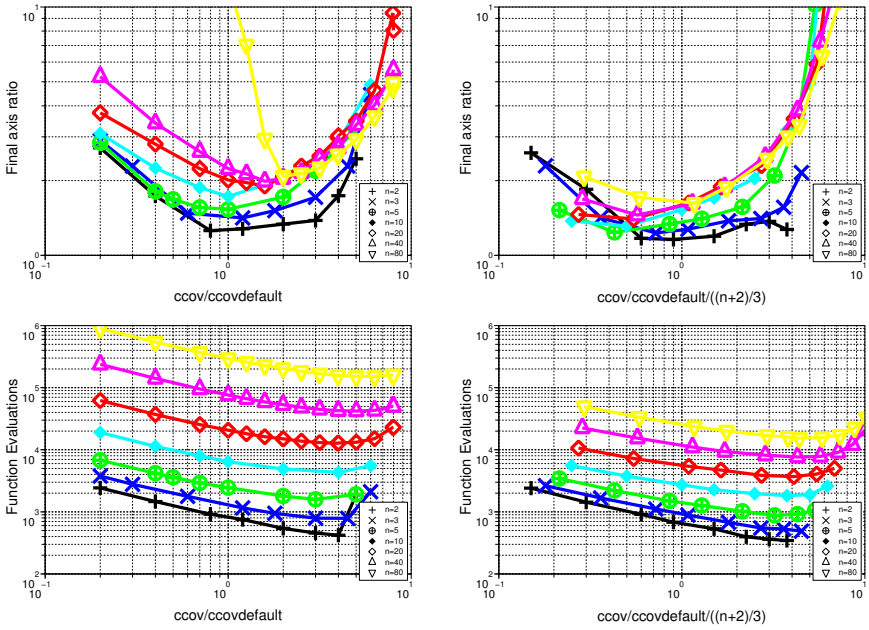
Figure 1: Results of default CMA-ES and sep-CMA-ES on the sphere function with initial covariance matrix $\boldsymbol{C}^{(0)}$ set to $\boldsymbol{I}$ for problem dimensions going from 2 to 40. A point is shown if, out of the 21 runs done for each set-up, all runs succeed in reaching the target function value of 10^{-9}. The top subfigures show the median axis ratio of the final covariance matrix whereas the bottom subfigures show the mean number of function evaluations.

4.1 Test Functions

All the functions are described in Table 1. We introduce the block-rotated ellipsoid function, $f^{\beta,m}_{\text{blockelli}}$. It is the compound of of the axis-parallel ellipsoid function with a transformation defined by an $n \times n$ matrix of m identical blocks along its diagonal, each of these blocks being a rotation matrix of size n/m. The block-rotated ellipsoid function allows us to control the separability of the problem. The transition can operate smoothly from the case where the number of blocks is n, which is equivalent to the axis-parallel ellipsoid function, to the case where m is equal to 1 and it is equivalent to the ellipsoid function.

Another variant of the ellipsoid function, $f_{\text{hyperelli}}$ used in [8], is tested, differing from the ellipsoid function by the fact that it is more biased toward the n-th component. Another function we considered is the well-known Rosenbrock function, f_{Rosen}, which is non-convex, not unimodal for larger dimensions and non separable. We tested the Rosenbrock function using its rotated version as well.

The Sum of different powers (Diff-Pow) function, f_{diffpow}, sums the different components of the argument to different powers and is unimodal. The axis-parallel Diff-Pow function is separable contrarily to the Diff-Pow function.



Default CMA-ES sep-CMA-ES

Figure 2: Results of default CMA-ES and sep-CMA-ES on the sphere function, initialising the covariance matrix $\boldsymbol{C}^{(0)}$ to a diagonal matrix with diagonal elements $(1, \beta^{\frac{1}{n-1}}, \beta^{\frac{2}{n-1}}, \ldots, \beta)$ for problem dimensions going from 2 to 40. A point is shown if, out of the 21 runs done for each set-up, all runs succeed in reaching the target function value of 10^{-9}. The top subfigures show the median axis ratio of the final covariance matrix whereas the bottom subfigures show the mean number of function evaluations. No success is observed for $c_{\text{cov}} = 0$.

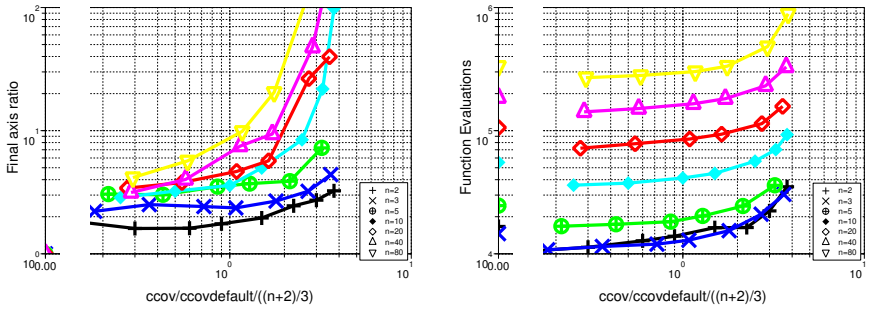


Figure 3: Results of sep-CMA-ES on the ellipsoid function ($\beta = 10^3$). A point is shown if, out of the 21 runs done for each set-up, all runs succeed in reaching the target function value of 10^{-9}. The subfigure on the left shows the median axis ratio of the final covariance matrix whereas the subfigure on the right shows the mean number of function evaluations.

Table 1: Test functions used for performance. For all experiments on all functions, the initialisation range is $[-20, 80]^n$ and initial step-size is $\sigma^{(0)} = \frac{100}{3}$. For the Rosenbrock, the Diff-Pow and the hyper-ellipsoid functions, $\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$, where $\boldsymbol{Q}$ is either $\boldsymbol{I}$ or an orthogonal $n \times n$ matrix with each column vector $\boldsymbol{q_i}$ being a uniformly distributed unit vector implementing an angle-preserving transformation. For the block rotated ellipsoid function, the orthogonal $n \times n$ matrix $\boldsymbol{Q}$ is either equal to $\boldsymbol{I}$ in the case of the axis-parallel function or equal to a block diagonal matrix with an orthogonal matrix $\boldsymbol{P}$ implementing an angle-preserving transformation repeated m times along its diagonal.

Name	Function	f_{target}
Rotated Rosenbrock	$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} 100\,(y_i^2 - y_{i+1})^2 + (y_i - 1)^2$	10^{-9}
Rotated Diff-Pow	$f_{\text{diffpow}}(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert y_i \rvert^{i+1}$	10^{-14}
Block-rotated Ellipsoid	$f^{\beta,m}_{\text{blockelli}}(\boldsymbol{x}) = f^{\beta}_{\text{elli}}(\boldsymbol{y})$	10^{-9}
Hyper-ellipsoid	$f_{\text{hyperelli}}(\boldsymbol{x}) = \sum_{i=1}^{n} (i\ y_i)^2$	10^{-10}

4.2 Experimental Set-up

All the algorithm parameters of the sep-CMA-ES are set to their default value except for the learning rate. First, we will show the behaviour of the sep-CMA-ES algorithm dealing with the typical optimisation task on the axis-parallel ellipsoid function.

CPU-time Experiments The amount of CPU-time for the algorithms to reach a number of function evaluations is measured while the dimensionality of the problem varies: n ranges from 10 to 5120. We have implemented the sep-CMA-ES algorithm from the purecmaes.m Matlab code[3].

We compare the sep-CMA-ES with variants of the CMA-ES algorithm that will postpone the eigendecomposition until after $(c_{\text{cov}} n)^{-1}\alpha$ generations, $\alpha \in \{0, 0.1, 1\}$. These variants have also been implemented from the purecmaes.m code. We make sure the number of function evaluations is large enough so that the eigendecomposition is computed at least 10 times: 5×10^4 function evaluations when α is equal to 0.1 or 1 and the dimension is larger than 320, 10^4 otherwise. Three trials are done for each algorithm on each dimensionality. We test using two population sizes: $\lambda = 4 + \lfloor 3 \ln n \rfloor$ and $\lambda = 2n$. Experiments were performed on a single (no hyper-threading) Intel Core 2 processor 2.66GHz with 2GB RAM.

Performance Experiments We use the Scilab version of the sep-CMA-ES and CMA-ES algorithms. All the functions are tested with n ranging from 2 to 1024 at most.

[3]http://www.bionik.tu-berlin.de/user/niko/purecmaes.m

For all problems, the starting point $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ is chosen in $[-20, 80]^n$ and the initial step-size $\sigma^{(0)}$ is one third of the interval width. If the target function value given in Table 1 is reached within 10^7 function evaluations, a run is considered successful. Performances are averaged over 11 runs for the lower dimensions ($n < 100$), 2 runs for larger dimensions. The rotation matrix $\boldsymbol{Q}$ is changed for every single run. The elementary block in the block-rotation involved in the block-rotated ellipsoid function is generated as for the rotation matrix previously mentioned.

In addition to the comparison of sep-CMA-ES to CMA-ES, we will also examine previously published results: the ES algorithm with derandomised mutative step-size control [8] denoted as indi-ES in the following, the AII-ES [5], the MVA-ES [9] and the L-CMA-ES [7]. We use the same starting point, initial step-size (where applicable) and same population sizes as those described in each of these cited works.

5 Results and Discussion

The dynamic behaviour of the sep-CMA-ES algorithm is studied first. In Fig. 4, the function value decrease steeply in the process of optimisation, even steeper after 3000 function evaluations when the adaptation of the larger eigenvalues has been done (cf. bottom-left subfigure). Other observations are that: (i) the step-size decreases as expected, (ii) the diagonal elements of the covariance matrix all go to zero as well, (iii) the axis ratio behaves correctly since its value is close to the square root of the condition number and is approximatively to 10^3 after 3000 function evaluations. Since the sep-CMA-ES algorithm shows sound results, we proceed to the CPU-time experiments.

5.1 CPU-time Experiments

We display the CPU-time per function evaluations versus the dimension for CMA-ES and sep-CMA-ES in Fig. 5. For the default population size (top figure), the time complexity of sep-CMA-ES scales like $n^{1.2}$ in the larger dimensions. In this context, if the eigendecomposition in CMA-ES is done at each iteration ($\alpha = 0$), the time complexity scales like $n^{2.7}$. The use of outdated covariance matrices reduces the time complexity to $n^{1.8}$, $n^{1.9}$ for $\alpha = 0.1, 1$ respectively, but sep-CMA-ES still outperforms CMA-ES by a factor of at least 6 for $n = 100$.

Experiments for $\lambda = 2n$ show that sep-CMA-ES achieves a linear time complexity for the larger dimensions. Again it clearly outperforms CMA-ES by a factor of 10 when $n = 100$, even for $\alpha = 1$ for which the time complexity scales with $n^{1.8}$ for dimensions up to 1000.

5.2 Performance Experiments

On Separable Functions On the axis-parallel ellipsoid (Fig. 6) and Diff-Pow (Fig. 7) functions, the sep-CMA-ES outperforms CMA-ES by a factor of 10 for problems of dimension 100, the gap of performances widening as the dimension increases. The sep-CMA-ES algorithm using a faster learning rate yields good results compared to those of CMA-ES on separable functions. Ill-conditioning affects sep-CMA-ES less than CMA-ES as the results of sep-CMA-ES worsens

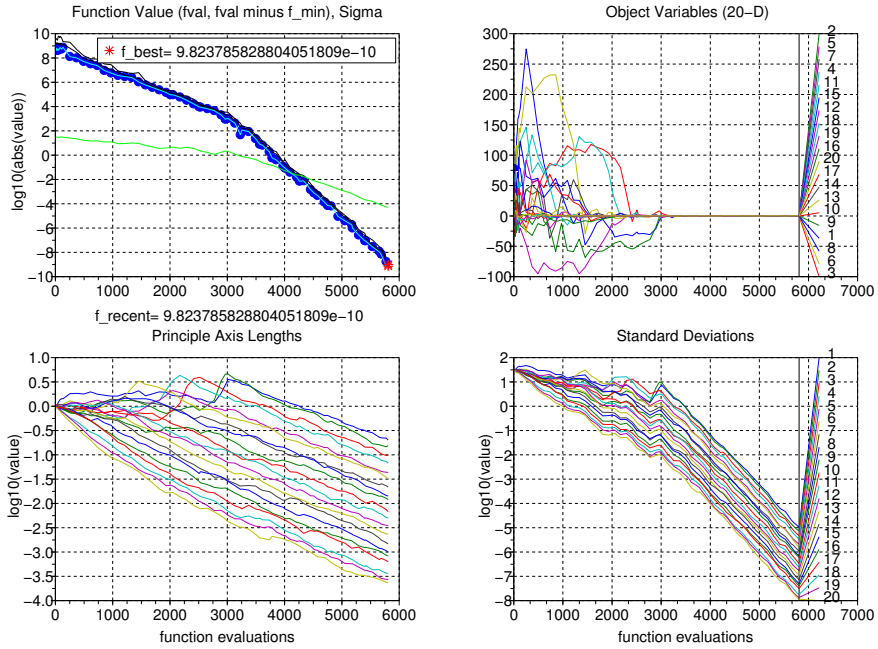


Figure 4: Single run of sep-CMA-ES on the axis-parallel ellipsoid function (the condition number is 10^6, the dimension is 20).

by a factor of around 2 when comparing results on condition number 1 and 10^{10}, whereas the performance of CMA-ES degrades by a factor of at most 60.

Rotated Diff-Pow and Block-Rotated Ellipsoid Functions The performance of sep-CMA-ES deteriorates on these non axis-parallel functions (Fig. 7 and 6): on the rotated Diff-Pow function, no runs reached the target function value in 10^7 function evaluations. The performance of sep-CMA-ES on the (1 block) ellipsoid function quickly worsens as the condition number increases, scaling linearly with the condition number. For a given condition number when n varies, the sep-CMA-ES algorithm performs gradually better on the block-rotated ellipsoid function as the number of blocks increases. For a condition number $\beta = 10^6$, we can observe (i) that the sep-CMA-ES performs the same whether there are 8 or n blocks, the two curves being close to indistinguishable, (ii) and the sep-CMA-ES performs better than the CMA-ES algorithm when there are more than 2 blocks.

On the Rosenbrock Function The sep-CMA-ES succeeds in optimising the Rosenbrock function. It is more effective than on the rotated Rosenbrock function by a factor of 10 at most in the larger dimensionalities. In the comparison with the CMA-ES, sep-CMA-ES does not perform better on the Rosenbrock function in the smaller dimensionalities and on the rotated Rosenbrock on all of the dimensions we tested. On the Rosenbrock function for the larger dimensions though ($n > 100$), the search costs, *i.e.* the number of function evaluations to reach a target function value) of the sep-CMA-ES is lower by a factor of about

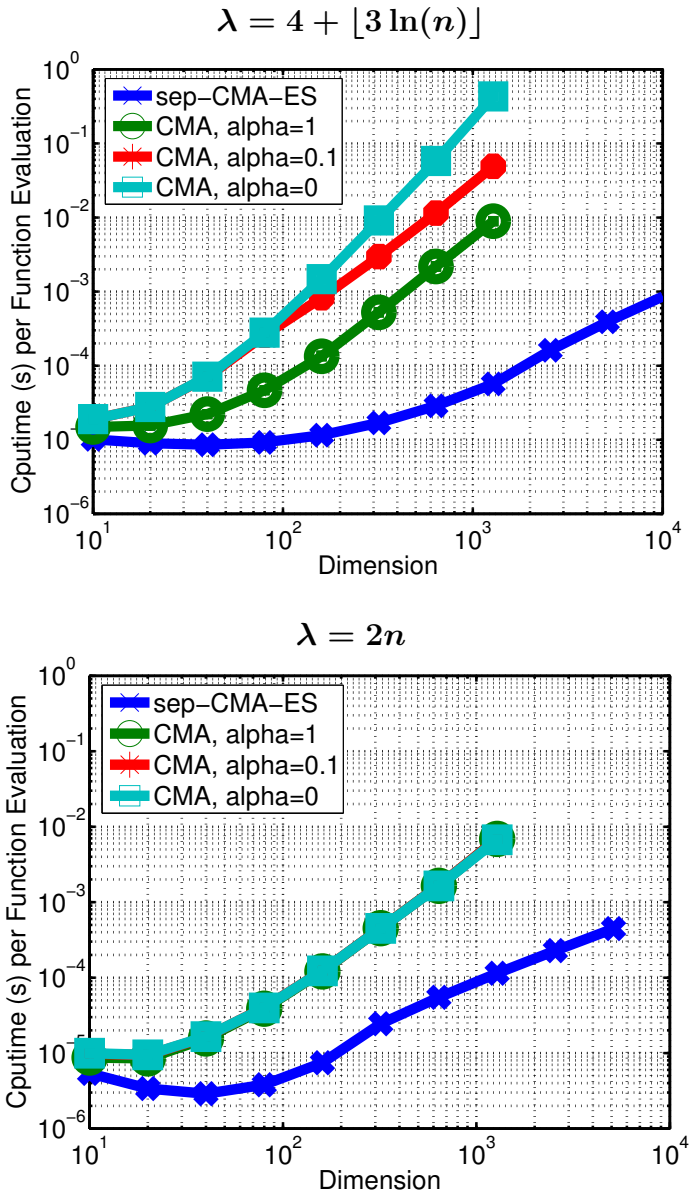


Figure 5: Timing results of default CMA-ES and sep-CMA-ES on the axis-parallel ellipsoid function. Both algorithms are tested for two different population sizes λ. Different variants of the CMA-ES algorithm are tested: the eigendecomposition of the covariance matrix is postponed until $(c_{\text{cov}}n)^{-1}\alpha$. For all experiments, the processing time was measured on 3 trials of 10^4 evaluations each. Lines show median of the distribution, vertical error-bars show minimum and maximum duration divided by the number of function evaluations (all indistinguishable).

one half at best. This surprising result can be interpreted as sep-CMA-ES having a faster learning rate than CMA-ES resulting in better performances.

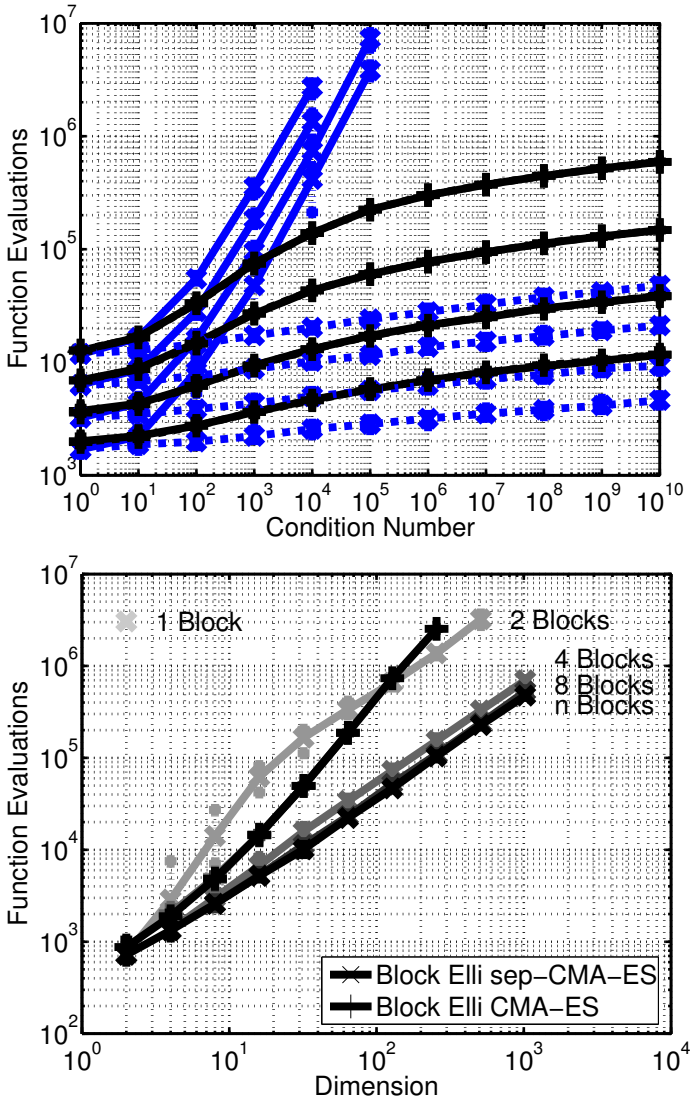


Figure 6: Results of sep-CMA-ES (x) on the block-rotated (1, 2, 4, 8, n blocks) ellipsoid functions, compared to CMA-ES (+). Top subfigure shows results on the 1 block (plain lines) and n blocks ellipsoid (dashed lines) with condition number β from 1 to 10^{10} in dimension 10 (bottom most line), 20, 40, 80 (top most line). Bottom subfigure shows results when the dimension varies. Lines show median of the distribution, vertical error-bars show minimum and maximum number of evaluations on successful runs out of 11 runs on smaller dimension ($n < 100$), 2 runs otherwise.

By multiplying the obtained search costs by the CPU-time per function evaluations we can estimate the overall CPU-time of the optimisation of the Rosenbrock function resulting in the bottom subfigure of Fig. 8. It is an opti-

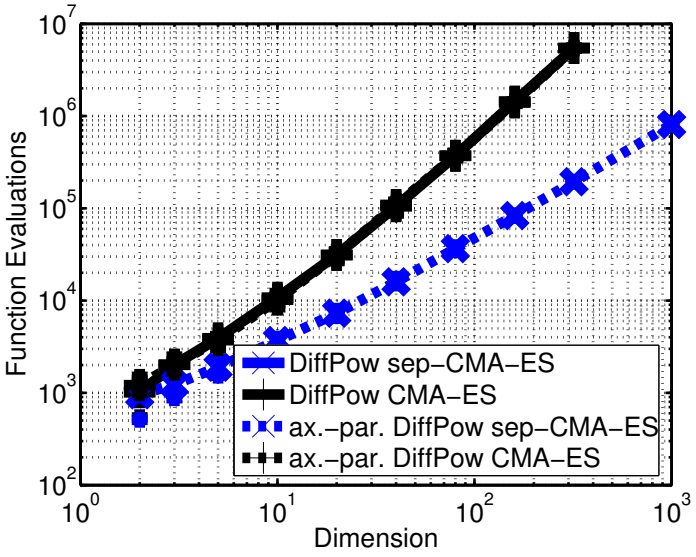


Figure 7: Results of sep-CMA-ES on the Diff-Pow function, compared to CMA-ES. The function is tested in its axis-parallel version (dashed lines) and in its rotated version (plain lines). Lines show median of the distribution, vertical error-bars show minimum and maximum number of evaluations on successful runs out of 11 runs on smaller dimension ($n < 100$), 2 runs otherwise.

mistic estimation of the duration of the task that is provided in this subfigure since it assimilates the CPU-time of the evaluation of the Rosenbrock function to that of the axis-parallel ellipsoid function. This approximation should shift the intersection of the two curves but not change the aspect of the subfigure: the sep-CMA-ES algorithm is expected to solve the Rosenbrock function more effectively than CMA-ES when the dimension is around 10^2 and above.

Comparison to Other Algorithms In the comparison in Table 2 with indi-ES and in Table 3 with AII-ES, we can see that sep-CMA-ES performs as well, otherwise better than the two simpler ES on separable functions such as the axis-parallel ellipsoid, hyper-ellipsoid and Diff-Pow functions. The gain obtained from having a larger population size than for the ES is equal to about 20% for the ellipsoid function, almost 50% for the Diff-Pow function) since we also tested sep-CMA-ES with population size (1, 10)-selection.

Comparisons in Table 3 and 4 show that sep-CMA-ES essentially does better than both MVA-ES and L-CMA-ES, though on the (rotated) ellipsoid, these two algorithms will perform the same as on the axis-parallel ellipsoid whereas the performance of sep-CMA-ES degrades.

On the Rosenbrock function, in the dimensions considered, sep-CMA-ES does not perform well. A (1, 10)-selection policy is better than $(\mu/\mu_W,\ \lambda)$ by a factor of about 30%.

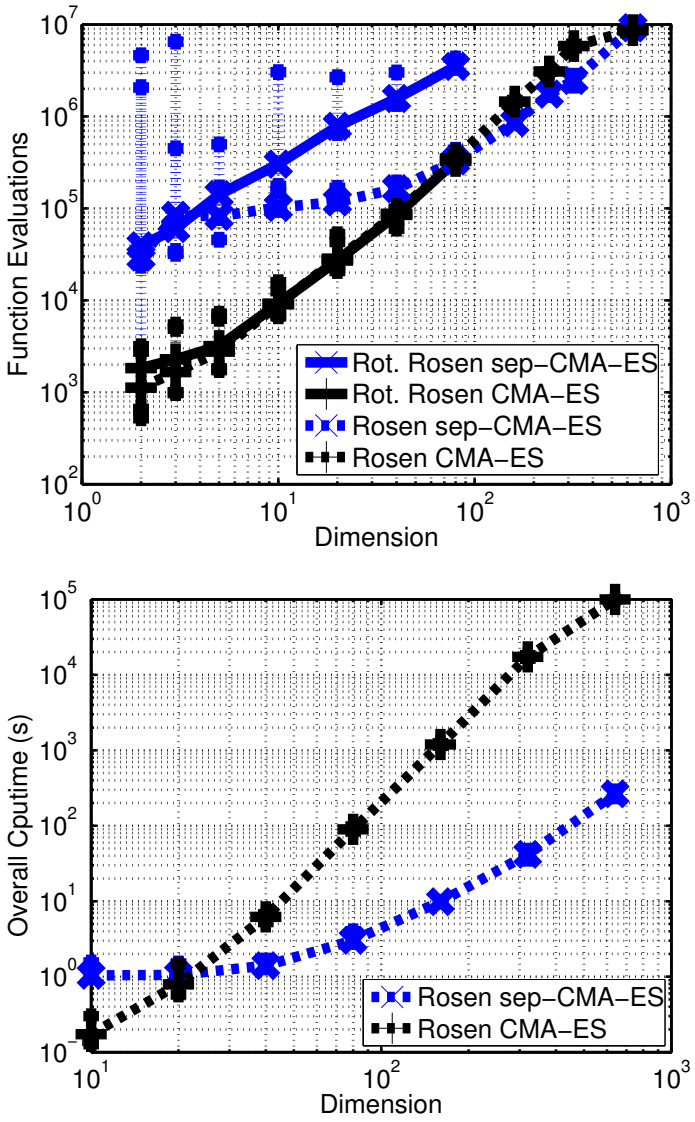


Figure 8: Results of sep-CMA-ES on the Rosenbrock function (in dashed line, rotated Rosenbrock function in plain line) compared to CMA-ES. Lines show median of the distribution, vertical error-bars show minimum and maximum number of evaluations on successful runs out of 11 runs on smaller dimension, 2 runs on larger dimension. Target function value is 10^{-9}.

6 Summary and Conclusion

We presented a simple modification of the CMA-ES that reduces the $n + \frac{n^2-n}{2}$ strategy parameters of the original algorithm to the n diagonal components of the covariance matrix, resulting in sep-CMA-ES. The sep-CMA-ES is an algo-

Table 2: Number of function evaluations (the unit is 10^3 evaluations) to reach given target function value, f_{target}, plus-minus the standard deviation when available. The dimension is $n = 30$. The notation '(1, 10)-select.' refers to a population size of 10 with selection of the best individual at each generation. The initial values of the problem-independent parameters $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)}$ and the $\sigma^{(0)}$ (where applicable) are given in the table. The results shown with standard deviation are averaged over 3 runs.

Algorithm	Population	$f_{\text{hyperelli}}$ axis-parallel $f_{\text{target}} = 10^{-10}$ $\sigma^{(0)} = 1$ $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = (1, ..., 1)^T$	f_{Rosen} axis-parallel $f_{\text{target}} = 10^{-6}$ $\sigma^{(0)} = 0.1$ $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = \mathbf{0}$	f_{diffpow} axis-parallel $f_{\text{target}} = 10^{-20}$ $\sigma^{(0)} = 1$ $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = (1, ..., 1)^T$
indi-ES [5]	(1, 10)-select.	6.6	80	9.7
sep-CMA-ES	(1, 10)-select.	7.2 ± 4%	81 ± 2%	19 ± 3%
CMA-ES	$(7/7_W,\ 14)$	13 ± 3%	**45 ± 2%**	79 ± 4%
sep-CMA-ES	$(7/7_W,\ 14)$	**5.9 ± 4%**	106 ± 3%	**9.6 ± 3%**

Table 3: Number of function evaluations (the unit is 10^3 evaluations) to reach given target function value, $f_{\text{target}} = 10^{-9}$, plus-minus the standard deviation when available, $n = 20$. The results shown with standard deviation are averaged over 3 runs. No success was observed for MVA-ES on the ellipsoid function (in 3.5×10^5 function evaluations).

Algorithm	Population	f_{elli} axis-parallel $\sigma^{(0)} = 1$ (when possible) $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = (1, \ldots, 1)^T$	f_{Rosen} axis-parallel $\sigma^{(0)} = 0.1$ (when possible) $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} = \mathbf{0}$
AII-ES [5]	(1, 10)-select.	12	**21**
MVA-ES [9]	(1, 10)-select.	no success (maxevals 3.5×10^5)	57 ± 50 % (Min-max range, 70 runs)
MVA-ES [9]	(5, 35)	no success (maxevals 3.5×10^5)	78 ± 45 % (Min-max range, 70 runs)
CMA-ES rank-1	(1, 10)-select.	27 ± 2%	26 ± 7%
CMA-ES rank-1	(5, 35)	43 ± 2%	37 ± 2%
CMA-ES rank-μ	$(6/6_W,\ 12)$	20 ± 0.6%	**21 ± 3%**
sep-CMA-ES	(1, 10)-select.	7.6 ± 2%	78 ± 3%
sep-CMA-ES	$(6/6_W,\ 12)$	**5.4 ± 2%**	116 ± 1%

rithm with individual variances and independent sampling which can exploit a large population size. The learning rate of the algorithm was carefully identified to adjust to the reduction of the model complexity.

Table 4: Number of function evaluations (the unit is 10^3 evaluations) to reach given target function value, $f_{\text{target}} = 10^{-14}$, plus-minus the standard deviation when available, $n = 40$. The results shown with standard deviation are averaged over 3 runs.

Algorithm	f_{elli} axis-parallel $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} \in [-5, 5]^n$ $\sigma^{(0)} = 5$	f_{Rosen} axis-parallel $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)} \in [-2, 2]^n$ $\sigma^{(0)} = 2$
L-CMA-ES ($m = 5$) [7]	2000	54
L-CMA-ES ($m = 15$) [7]	706	63
CMA-ES rank-1 [7]	73	63
CMA-ES rank-μ	$45 \pm 0.8\%$	$\mathbf{51 \pm 5\%}$
sep-CMA-ES	$\mathbf{11 \pm 0.5\%}$	$191 \pm 2\%$

The sep-CMA-ES algorithm has a linear time and space complexity. It does not learn the interdependencies of the objective function which favours sep-CMA-ES on separable functions. The assets of sep-CMA-ES allow for solving black-box optimisation tasks even with large problem dimensionality. On separable functions sep-CMA-ES performs better than CMA-ES and other variants that scale linearly.

Results from our experiments on the non-separable Rosenbrock function came as a surprise: for larger dimensions sep-CMA-ES outperforms CMA-ES. This was observed as well on the newly introduced block-rotated ellipsoid function.

From these results, we state simple consequence for a policy: first sep-CMA-ES is used for only a fraction of the learning time of CMA-ES, then one switches to CMA-ES (by just using a different update rule and changing the learning rate). Unfortunately this will be still significantly inferior in some cases like the Cigar function (where sufficient dependencies can be learnt in linear time only with CMA) or the separable Diff-Powers function (where the scaling continuously varies and the fast learning rate of sep-CMA would be beneficial for more than just a fraction of the learning time).

Using sep-CMA might improve the comparatively slow performance in the very beginning of an optimisation run of CMA-ES which we ascribe, at least partly, to the fact that overall step-size and in particular coordinate scaling cannot decrease as fast in CMA-ES as in many other evolutionary algorithms. Insights from this work will serve as a baseline comparison and a stepstone to constructing variants of CMA with linear time and space complexity.

References

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

[2] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Parallel Problem Solving*

from Nature - PPSN VIII, LNCS 3242, pages 282–291. Springer, 2004.

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

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

[5] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. J. Eshelman, editor, *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1995.

[6] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 453–460, New York, NY, USA, 2006. ACM.

[7] J. N. Knight and M. Lunacek. Reducing the space-time complexity of the cma-es. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 658–665, New York, NY, USA, 2007. ACM.

[8] A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size Adaptation Based on Non-local Use of Selection Information. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pages 189–198. Springer-Verlag, LNCS 866, 1994.

[9] J. Poland and A. Zell. 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, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1050–1055, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

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