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

Raymond Ros, Nikolaus Hansen

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

Raymond Ros[1] and Nikolaus Hansen[2]

[1] Univ. Paris-Sud, LRI, UMR 8623 / INRIA Saclay, projet TAO, F-91405 Orsay, France.
Raymond.Ros@lri.fr
[2] Microsoft Research–INRIA Joint Centre, 28 rue Jean Rostand, 91893 Orsay Cedex, France.
Nikolaus.Hansen@inria.fr

Abstract. This paper proposes a simple modification of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for high dimensional objective functions, reducing 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 dimensions larger than a hundred, even on the non-separable Rosenbrock function, the sep-CMA-ES needs fewer function evaluations than CMA-ES.

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. Difficult real parameter optimisation problems also exhibit essential dependencies between the parameters, and learning these dependencies has been successfully addressed by covariance matrix adaptation (CMA) [2, 4]. The CMA learns all pair-wise dependencies between all parameters by updating a covariance matrix for the sample distribution. The CMA was originally introduced for evolution strategies (ESs) but recently applied also in Evolutionary Gradient Search [1]. 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 to reach a target function value 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.

Motivation A principle limitation of CMA results from the degrees of freedom, $\frac{n^2+n}{2}$, in the covariance matrix, also referred to as strategy parameters. The full learning task scales roughly with n^2 (see *e.g.* [4]) and can dominate the search costs (in this case, the learning phase is much longer than the convergence phase). A second limitation lies in the *internal* computational complexity. (i) 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. (ii) Updating the covariance matrix has a complexity of $(\mu + 1)n^2$ since the so-called rank-μ update [3] amounts to μ covariance matrix updates. (iii) 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 $\boldsymbol{C}$. Usually, this computation is postponed until after $n/10$ generations and slightly outdated distributions are sampled [4]. Consequently, the complexity of this step becomes n^2 per generation.[3] In conclusion, several steps in the CMA algorithm have a computational complexity of $\Theta\left(n^2\right)$.

The most obvious option toward improving the scaling behaviour for the search costs is to *reduce the degrees of freedom* in the covariance matrix. We think of several ways to reduce the degrees of freedom, resulting in a family of potentially useful modifications of CMA-ES which *trade off model complexity for learning speed*. As long as the model complexity remains sufficient, search costs decrease because of a reduced learning period. In this paper, we pursue the arguably simplest modification of CMA that reduces the degrees of freedom in the covariance matrix to n. Even though we interpret this modification, sep-CMA, rather as a preliminary step, it reveals some interesting, surprising and promising perspectives on its own.

Previous Works on Favourably Scaling CMA Variants Some ESs, which were introduced prior to CMA-ES, implement key features of the CMA-ES and scale linearly with the dimension. In [8], a $(1, \lambda)$-ES with cumulation for individual step-size adaptation is proposed.[4] An extension of this derandomised step-size adaptation, denoted AII-ES in [5], combines this individual step-size adaptation with the adaptation of one direction, overall updating $2n$ strategy parameters.

The MVA-ES algorithm [9] adapts one main (mutation) vector. The time complexity of the algorithm is n according to the size of the main vector. The MVA-ES is efficient in the specific case of objective functions with a single preferred mutation direction. In L-CMA-ES [6] a parameter m allows to control the dimensionality of the representation of the mutation distribution. The learning is restrained to $m \leq n$ main components. For the two extremes, if $m = 1$, L-CMA-ES is somewhat similar to MVA-ES and if $m = n$, it is equivalent to the original CMA-ES.

In this paper, we address another subspace of strategy parameters that can be easily identified: the diagonal of the covariance matrix.

[3] 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.

[4] The algorithm is very similar to $(1, \lambda)$-sep-CMA-ES.

Objectives of this Paper We address two main objectives. (i) Formulating a smallest possible modification of CMA, denoted as sep-CMA, that can learn a scaling of variables in linear time. The sep-CMA-ES is, to our knowledge, the first derandomised evolution strategy with linear time complexity that can exploit a large population effectively, just as the CMA-ES. (ii) Comparing the performance of sep-CMA-ES on both separable *and non-separable* functions to CMA-ES and other previously proposed evolutionary algorithms. Surprisingly, sep-CMA-ES will turn out to be advantageous not only on separable, but also on significantly non-separable functions.

The remainder of this paper is organised as follows. Section 2 will introduce sep-CMA-ES, derived from the original CMA-ES. In Section 3, test functions and the test set-up are given. Results from the experiments are presented in Section 4 and provide insights from which conclusions are drawn in the last section.

2 sep-CMA-ES

We begin by presenting the CMA-ES algorithm, introduced in [4]. The $(\mu/\mu_W, \lambda)$-CMA-ES is described in **Alg. 1**. The description closely follows [2] in using weighted recombination of offspring along with a rank-μ update of the covariance matrix such that a large population size can be exploited.

For sep-CMA-ES, two simple changes are undertaken in the original CMA-ES. (i) The covariance matrix $\boldsymbol{C}$ is in effect constrained to be diagonal, (ii) the learning rate c_{cov} is increased. When the covariance matrix is diagonal, the mutation distribution is sampled independently in the given coordinate system using n individual variances. The only modification in the CMA-ES appears in line 11 which is modified to be:

$$\boldsymbol{D} = \sqrt{\text{diag}(\boldsymbol{C})} \tag{1}$$

where diag$(\boldsymbol{C})$ is a diagonal matrix with the same diagonal elements as $\boldsymbol{C}$. The matrix $\boldsymbol{B}$ remains $\boldsymbol{I}$ for all iterations.

Because only the diagonal elements of the covariance matrix are utilized, only the diagonal of the covariance matrix must be updated in line 9 and the time complexity of this step becomes linear in n. All other steps in the algorithm become at most linear, because $\boldsymbol{B} = \boldsymbol{I}$ can be removed from the equations.

In contrast to the CMA-ES, the sep-CMA-ES is not rotationally invariant. The degrees of freedom in the covariance matrix reduce from $n + \frac{n^2-n}{2}$ to n, thus the learning rate in line 9 can be increased. Tests on standard functions using different values for the learning rate c_{cov} were done in [10]. For obtaining a similar behaviour than that of CMA-ES when c_{cov} varies, the learning rate for sep-CMA-ES had to be multiplied by $\frac{n+2}{3}$. In all following experiments, $c_{\text{cov}} = \frac{n+2}{3} c_{\text{cov}}^{\text{def}}$ is used for sep-CMA-ES.

3 Test Functions and Methods

Test Functions All test functions are given in Table 1. We introduce the block-rotated ellipsoid function, $f_{\text{blockelli}}^{\beta,m}$. It is the compound of an axis-parallel ellipsoid function with an $n \times n$ matrix $\boldsymbol{Q}$ with m identical orthogonal matrices of size $\frac{n}{m} \times \frac{n}{m}$ along its

Parameter Setting:

$$\lambda = 4 + \lfloor 3\ln(n)\rfloor,\ \mu = \lfloor \tfrac{\lambda}{2}\rfloor,\ w_i = \frac{\ln(\mu+1)-\ln(i)}{\sum_{j=1}^{\mu}\ln(\mu+1)-\ln(j)}\ (i=1,\ldots,\mu),\ \mu_{\mathrm{w}} = \frac{1}{\sum_{i=1}^{\mu} w_i^2},$$
$$c_\sigma = \frac{\mu_{\mathrm{w}}+2}{n+\mu_{\mathrm{w}}+3},\ d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\mathrm{w}}-1}{n+1}} - 1\right) + c_\sigma,$$
$$c_{\mathrm{c}} = \frac{4}{n+4},\ \mu_{\mathrm{cov}} = \mu_{\mathrm{w}},\ c_{\mathrm{cov}} = c_{\mathrm{cov}}^{\mathrm{def}} = \frac{1}{\mu_{\mathrm{cov}}}\frac{2}{\left(n+\sqrt{2}\right)^2} + \left(1-\frac{1}{\mu_{\mathrm{cov}}}\right)\min\left(1, \frac{2\mu_{\mathrm{cov}}-1}{(n+2)^2+\mu_{\mathrm{cov}}}\right)$$

Initialisation: $g = 0, \boldsymbol{B} = \boldsymbol{I}, \boldsymbol{D} = \boldsymbol{I}, \boldsymbol{p_\sigma} = (0,\ldots,0)^T, \boldsymbol{p}_{\mathrm{c}} = (0,\ldots,0)^T, \boldsymbol{C} = \boldsymbol{I}$. The initial value of the parameters $\langle \boldsymbol{x}\rangle_{\mathrm{w}} \in \mathbb{R}^n$ and the step-size $\sigma \in \mathbb{R}$ is problem-dependent.
Repeat until a stopping criterion is reached:

```
1.   $g \leftarrow g + 1$
2.   $\boldsymbol{z_i} \sim \mathcal{N}(0, \boldsymbol{I})$ for $i = 1,\ldots,\lambda$
3.   $\boldsymbol{x_i} = \langle \boldsymbol{x}\rangle_{\mathrm{w}} + \sigma \boldsymbol{B}\boldsymbol{D}\boldsymbol{z_i}$
4.   $\langle \boldsymbol{x}\rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$ where $\boldsymbol{x}_{i:\lambda}$ denotes the $i$-th best individual out of the $\lambda$
5.   $\langle \boldsymbol{z}\rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}$ where $\boldsymbol{z}_{i:\lambda}$ denotes the $i$-th best mutation vector
6.   $\boldsymbol{p_\sigma} \leftarrow (1-c_\sigma)\boldsymbol{p_\sigma} + \sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_{\mathrm{w}}}\boldsymbol{B}\langle \boldsymbol{z}\rangle_{\mathrm{w}}$
7.   $H_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p_\sigma}\|}{\sqrt{1-(1-c_\sigma)^{2g}}} < (1.4 + \frac{2}{n+1})E(\|\mathcal{N}(0,\boldsymbol{I})\|) \\ 0 & \text{otherwise (stalling the update of } \boldsymbol{p}_{\mathrm{c}} \text{ if } \boldsymbol{p_\sigma} \text{ is large)} \end{cases}$
8.   $\boldsymbol{p}_{\mathrm{c}} \leftarrow (1-c_{\mathrm{c}})\boldsymbol{p}_{\mathrm{c}} + H_\sigma\sqrt{c_{\mathrm{c}}(2-c_{\mathrm{c}})}\sqrt{\mu_{\mathrm{w}}}\boldsymbol{B}\boldsymbol{D}\langle \boldsymbol{z}\rangle_{\mathrm{w}}$
9.   $\boldsymbol{C} \leftarrow (1-c_{\mathrm{cov}})\boldsymbol{C} + \frac{1}{\mu_{\mathrm{cov}}}c_{\mathrm{cov}}\boldsymbol{p}_{\mathrm{c}}(\boldsymbol{p}_{\mathrm{c}})^T$
         $+ c_{\mathrm{cov}}\left(1-\frac{1}{\mu_{\mathrm{cov}}}\right)\sum_{i=1}^{\mu} w_i \boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_{i:\lambda}(\boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_{i:\lambda})^T$
10.  $\sigma \leftarrow \sigma\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p_\sigma}\|}{E(\|\mathcal{N}(0,\boldsymbol{I})\|)} - 1\right)\right)$
11.  $[\boldsymbol{B}, \boldsymbol{D}^2] = \text{eigendecomposition}(\boldsymbol{C})$
```

Alg. 1: The CMA-ES algorithm; $=$ and $\leftarrow$ denote left-hand assignments. Components in $\mathcal{N}(0,\boldsymbol{I}) \in \mathbb{R}^n$ are independent (0,1)-normally distributed. eigendecomposition$(\boldsymbol{C})$ returns normalized, orthogonal eigenvectors as columns of $\boldsymbol{B}$ and the respective eigenvalue square roots as diagonal elements of $\boldsymbol{D}$. To achieve sep-CMA-ES, line 11 is replaced by $\boldsymbol{D}^2 = \text{diag}(\boldsymbol{C})$ according to Eq. (1), and $c_{\mathrm{cov}} = \frac{n+2}{3}c_{\mathrm{cov}}^{\mathrm{def}}$

diagonal (m blocks). If the number of blocks $m = n$, the function is equivalent to the axis-parallel ellipsoid function, for $m = 1$ block, it is equivalent to the rotated ellipsoid function. The hyper-ellipsoid function, $f_{\text{hyperelli}}$, is used in [8] and is biased to more sensitive components than the ellipsoid function. We also use the well-known Rosenbrock function, f_{Rosen}, which is non-convex and non-separable. The rotated Rosenbrock function ($\boldsymbol{Q} \neq \boldsymbol{I}$) is tested as well. The sum-of-different-powers (Diff-Pow) function, $f_{\text{diffpow}}^{\beta}$, is unimodal and separable but reveals increasing differences in the sensitivities when approaching the optimum.

CPU-time Experiments The total CPU-time for a run with a given number of function evaluations is measured for different problem dimensions. For these experiments we have implemented the sep-CMA-ES and CMA-ES from the `purecmaes.m` Mat-

Table 1. Test functions. We have $\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$ and the orthogonal $n \times n$ matrix $\boldsymbol{Q}$ is either $\boldsymbol{I}$ for the axis-parallel case or, for the (fully) rotated function, an angle-preserving transformation generated according to [4]. For the block rotated ellipsoid function, $\boldsymbol{Q}$ equals to a block diagonal matrix with an orthogonal $\frac{n}{m} \times \frac{n}{m}$ matrix repeated m times along its diagonal

Name	Function		f_{target}
Rosenbrock	$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100\,(y_i^2 - y_{i+1})^2 + (y_i - 1)^2\right)$		10^{-9}
Diff-Pow	$f_{\text{diffpow}}^{\beta}(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert y_i \rvert^{2+\beta\frac{i-1}{n-1}}$,	$\beta = 10$ as default	10^{-14}
Block-Rotated Ellipsoid	$f_{\text{blockelli}}^{\beta,m}(\boldsymbol{x}) = \sum_{i=1}^{n} \beta^{\frac{i-1}{n-1}} y_i^2$,	$\beta = 10^6$ as default	10^{-9}
Hyper-ellipsoid	$f_{\text{hyperelli}}(\boldsymbol{x}) = \sum_{i=1}^{n} (i\, y_i)^2$		10^{-10}

lab code[5]. In the CMA-ES algorithm the eigendecomposition is postponed until after $\alpha(c_{\text{cov}}n)^{-1}$ generations, with α in $\{0, 0.1, 1\}$. The number of function evaluations for the time measurement is 5×10^4 when α is 0.1 or 1 and the dimension is larger than 320, otherwise 10^4, to make sure the eigendecomposition is computed at least ten times. Three trials are done for each algorithm on each dimension. Two population sizes are tested: $\lambda = 4 + \lfloor 3 \ln n \rfloor$ and $\lambda = 2n$. Experiments were performed on a single (no hyper-threading) Intel Core 2 processor 2.66GHz with 2GB RAM.

Performance Experiments We measure the number of function evaluations to reach the target function value from successful runs. For lower dimension ($n < 100$), 11 runs, otherwise 2 runs are conducted. If the target function value given in Table 1 is reached within 10^7 function evaluations, the run is considered successful. On the Rosenbrock function, at most 30%, usually less, of the runs per set-up converged to the local optimum with CMA-ES or sep-CMA-ES. These unsuccessful runs are disregarded in our performance analysis. The rotation matrix $\boldsymbol{Q}$ is changed for every single run, the same set of rotation matrices is used for testing both algorithms.

We use a Scilab version of the sep-CMA-ES and CMA-ES. For all problems, the starting point $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)}$ is chosen uniformly in $[-20, 80]^n$ and the initial step-size $\sigma^{(0)} = 100/3$ is one third of the interval width. In addition to the comparison of sep-CMA-ES to CMA-ES, we also compare to previously published results where we use the same starting point, initial step-size (when available) and population sizes as those described in each of the works cited below.

4 Results and Discussion

CPU-time Experiments **Figure 1** displays the total CPU-time divided by the number of function evaluations versus dimension for sep-CMA-ES and CMA-ES. For the default population size (left subfigure), sep-CMA-ES performs much faster. In larger dimensions, the time complexity of sep-CMA-ES empirically scales like $n^{1.2}$, the time

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

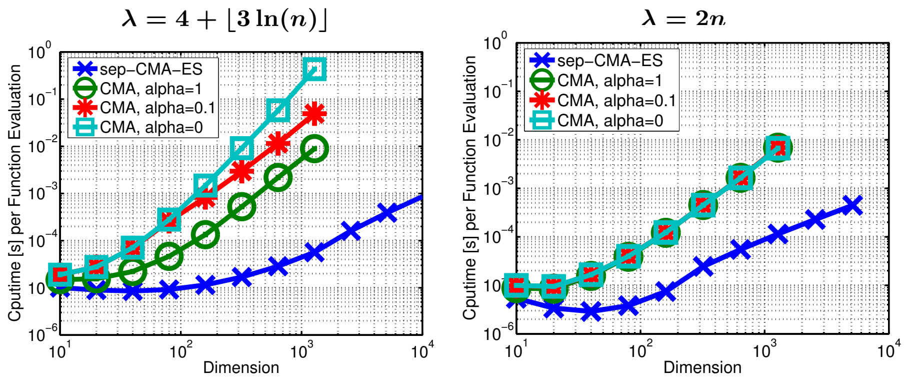

Fig. 1. CPU time per number of function evaluation for sep-CMA-ES (dark) and CMA-ES on the axis-parallel ellipsoid function for two different population sizes λ. The eigendecomposition of the covariance matrix in CMA-ES is postponed until after $(c_{\text{cov}}n)^{-1}\alpha$ generations. Lines show the median of three trials, vertical error-bars show minimum and maximum (all indistinguishable)

complexity of CMA-ES scales like $n^{2.7}$ if the eigendecomposition in CMA-ES is done at each iteration ($\alpha = 0$) and becomes slightly sub-quadratic if outdated covariance matrices are used ($\alpha = 0.1$ and 1), but sep-CMA-ES is still faster by a factor of at least six for $n = 100$ and at least a hundred for $n = 1000$. For $\lambda = 2n$ (right subfigure), sep-CMA-ES empirically achieves linear time complexity in larger dimensions whereas the time complexity of CMA-ES is quadratic. Again, sep-CMA-ES is clearly faster than CMA-ES by a factor of ten and forty for $n = 100$ and 1000 respectively.

Performance Experiments **Figure 2** shows the average number of function evaluations to reach f_{target} on different functions. On the separable Diff-Pow and ellipsoid function (left subfigures), the sep-CMA-ES outperforms CMA-ES by a factor of ten in 100-D and the performance gap widens as the dimension increases.

The performance of sep-CMA-ES deteriorates on the rotated functions: on the Diff-Pow function no run reached the target function value. On the block-rotated ellipsoid function, sep-CMA-ES does not succeed with 1 block (*i.e.* rotated ellipsoid function) except for $n = 2$. As the number of blocks increases, the sep-CMA-ES performs gradually better on the block-rotated ellipsoid function. For 2 blocks, sep-CMA-ES outperforms CMA-ES in dimensions larger than 200. Already for 4 blocks, where the condition number within the non-separable sub-problems equals to $10^{6/4} \approx 30$, sep-CMA-ES outperforms CMA-ES in all dimensions.

On the Rosenbrock function (non-rotated) the sep-CMA-ES is roughly 50 times slower than the CMA-ES in small dimensions (cf. right of Fig. 2). With increasing dimension the difference vanishes and, surprisingly, *sep-CMA-ES outperforms CMA-ES on the Rosenbrock function for dimension* $n > 100$ while the success rates remain close for both algorithms. This effect cannot be observed on the rotated Rosenbrock function, where sep-CMA-ES is outperformed by CMA-ES at least by a factor of ten up to 100-D.

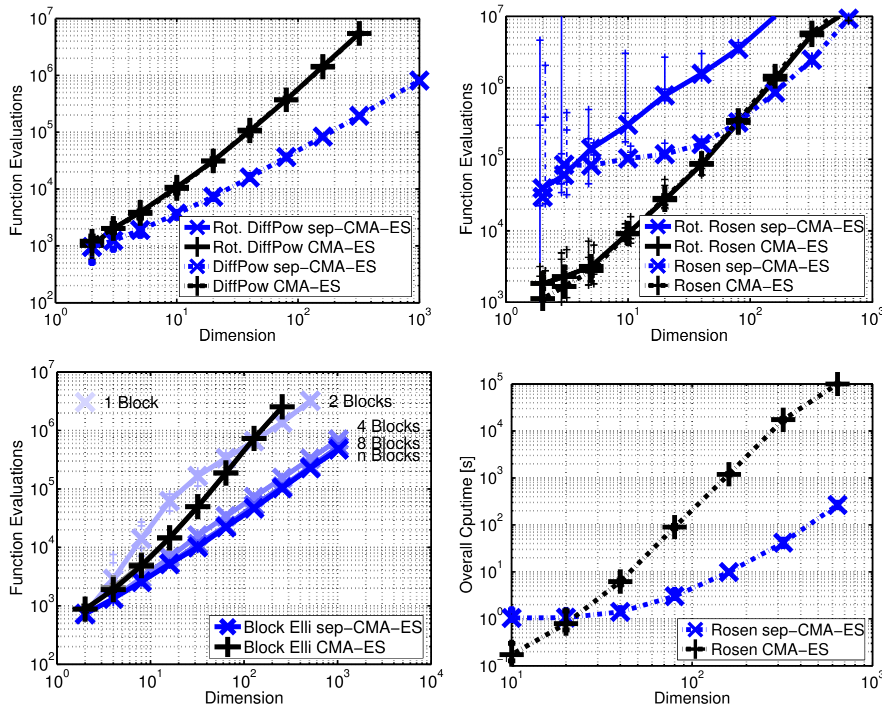

Fig. 2. Experimental results of sep-CMA-ES ($\times$) on the Diff-Pow function (top-left), the block-rotated ellipsoid (bottom-left) and Rosenbrock function (right), compared to CMA-ES (+). The functions are tested in their axis-parallel version (dashed lines) and in their rotated version (plain lines). Lines show median, vertical error-bars show quartiles of the number of evaluations of the successful out of 11 runs on smaller dimension ($n < 100$), out of 2 runs otherwise

The bottom-right of Fig. 2 investigates the advantage of sep-CMA-ES on cheap to evaluate objective functions. By multiplying the number of function evaluations needed on the Rosenbrock function by the CPU-time per function evaluation as given in Fig. 1, we obtain the *overall CPU-time* for optimising the non-separable Rosenbrock function (assuming the CPU-costs of the function evaluations are that of the axis-parallel ellipsoid function as used in the experiments from Fig. 1). The sep-CMA-ES becomes faster for dimensions larger than 20. In 100-D, sep-CMA-ES is already 50 times faster than CMA-ES and the gap widens with increasing dimension. The bottom-right subfigure of Fig. 2 is an optimistic scenario since the axis-parallel ellipsoid function is very cheap. More CPU-expensive functions will narrow the gap and shift the point where the two curves cross to the right, but never beyond $n = 100$.[6]

[6] The reason sep-CMA-ES becomes faster is not only because of its smaller time complexity but also because the search costs become lower.

Table 2. Mean number of function evaluations $[\times 10^3]$ to reach the target function value from 3 runs, plus-minus the standard deviation when available, $n = 30$. All functions are used in their non-rotated version. On the hyper-ellipsoid and Diff-Pow function, $\sigma^{(0)} = 1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = (1, \ldots, 1)^T$. On the Rosenbrock function, $\sigma^{(0)} = 0.1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = \mathbf{0}$

Function	f_{target}	indi-ES [8]	CMA-ES	sep-CMA-ES	
		(1, 10)-select.	$(7/7_W,\ 14)$	(1, 10)-select.	$(7/7_W,\ 14)$
$f_{\text{hyperelli}}$	10^{-10}	6.6	13±3%	7.2±4%	**5.9±4%**
f_{diffpow}^{n-1}	10^{-20}	9.7	79±4%	19±3%	**9.6±3%**
f_{Rosen}	10^{-6}	80	**45±2%**	81±2%	106±3%

Table 3. Mean number of function evaluations $[\times 10^3]$ to reach given target function value 10^{-9} from 3 runs, plus-minus the standard deviation when available, $n = 20$. In the case of MVA-ES, the range between maximum and minimum from 70 runs is displayed. All functions are used in their non-rotated version. On the ellipsoid function, $\sigma^{(0)} = 1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = (1, \ldots, 1)^T$. On the Rosenbrock function, $\sigma^{(0)} = 0.1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = \mathbf{0}$. No success was observed for MVA-ES on the ellipsoid function (in 3.5×10^5 function evaluations)

Function	AII-ES [5]	MVA-ES [9]		CMA-ES	sep-CMA-ES	
	(1, 10)-select.	(1, 10)-select.	(5, 35)	$(6/6_W,\ 12)$	(1, 10)-select	$(6/6_W,\ 12)$
f_{elli}	12	no success	no success	20±0.6%	7.6±2%	**5.4±2%**
f_{Rosen}	**21**	57±50%	78±45%	**21±3%**	78±3%	116±1%

Table 4. Mean number of function evaluations $[\times 10^3]$ to reach the target function value 10^{-14} from 3 runs, plus-minus the standard deviation when available, $n = 30$, $\lambda = 14$. All functions are used in their non-rotated version. On the ellipsoid function, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ is chosen randomly in $[-5, 5]^n$, $\sigma^{(0)} = 5$. On the Rosenbrock function, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ is chosen randomly in $[-2, 2]^n$, $\sigma^{(0)} = 2$

Function	L-CMA-ES [6, 7]		CMA-ES		sep-CMA-ES	
	rank-1, $m = 5$	rank-1, $m = 15$	rank-1	default	rank-1	default
f_{elli}	1900	700	46±0.4%	45±0.8%	**11±7%**	**11±0.5%**
f_{Rosen}	53	63	52±27%	**51±5%**	134±7%	191±2%

Comparison to Other Algorithms **Tables 2, 3 and 4** compare the performance of sep-CMA-ES with previous works. On separable functions (f_{elli}, $f_{\text{hyperelli}}$, f_{diffpow}), the sep-CMA-ES performs comparable to indi-ES [8] and AII-ES [5] (Tables 2 and 3) and greatly outperforms MVA-ES [9] and L-CMA-ES [6, 7] (Tables 3 and 4), while the latter are rotational invariant. On the Rosenbrock function, AII-ES and L-CMA-ES perform better than sep-CMA-ES, because they can learn a limited number of correlations.

5 Summary and Conclusion

We presented the sep-CMA-ES algorithm, a simple modification of the CMA-ES that reduces the $n + \frac{n^2-n}{2}$ degrees of freedom in the covariance matrix of the original algorithm to only n diagonal components, where n is the problem dimension. Consequently, in contrast to the CMA-ES, dependencies between variables are not captured and coordinates are sampled independently. Just like CMA-ES, the sep-CMA-ES can exploit a large population. The **advantages** of sep-CMA-ES are twofold: (i) it reduces the internal time and space complexity of CMA-ES from quadratic to linear. (ii) the learning rate for the covariance matrix can be increased by a factor of about $\frac{n}{3}$, considerably accelerating the adaptation of axis-parallel distribution ellipsoids.

Evaluation Results We evaluated sep-CMA-ES on separable and non-separable test functions by *measuring numbers of function evaluations*. Fully separable problems mainly served to confirm a proper implementation. On separable functions sep-CMA-ES significantly outperforms CMA-ES and the scale-up with the dimension is linear.

We introduced the **block-ellipsoid**, a convex-quadratic test function for which the "degree of non-separability" can be controlled. For low and moderate degrees of non-separability (relating to "non-separable condition numbers" of up to 30) the sep-CMA-ES outperforms CMA-ES *in all dimensions*, roughly by a factor of $\frac{n}{10} + 1$. For a larger degree of non-separability (relating to a non-separable condition number of 1000) an advantage can be observed only in larger dimension ($n > 100$). On the fully non-separable ellipsoid, CMA-ES is always far superior.

The well-known **Rosenbrock** function exhibits relevant dependencies between the variables posing no principle obstacle for sep-CMA-ES. In low dimensions, sep-CMA-ES is about 50 times slower than CMA-ES. The performance difference diminishes with increasing dimension and for dimensions $n > 100$, *sep-CMA-ES becomes faster than CMA-ES*. This effect can be attributed to the given coordinate system: on the *rotated* Rosenbrock function sep-CMA-ES never outperformed CMA-ES up to dimension 320.

Implications We perceive two principle benefits from sep-CMA-ES. First, the study of sep-CMA-ES allows to explicitly measure the benefits and drawbacks from learning dependencies. We can quantify the gain or loss that can be attributed to the ability to adapt the complete covariance matrix in CMA-ES. The sep-CMA-ES also allows insightful cross-comparisons with other "separable" algorithms (with only coordinate-wise operations). Second, the application of sep-CMA-ES to **real-world problems** will be advantageous (compared to CMA-ES) on high-dimensional objective functions, which either do not have too intricate dependencies between the decision variables (as it is the case for the Rosenbrock function) or are cheap to evaluate. In the first scenario, sep-CMA-ES needs fewer function evaluations if the adaptation of the scaling of variables helps to solve the function. The second scenario favors sep-CMA-ES, when the strategy internal time complexity becomes relevant, where CMA-ES is roughly $\frac{n}{10} + 1$ times slower. We presented natural examples for both scenarios, where sep-CMA-ES outperforms CMA-ES by a factor of about ten already in 100-D.

For combining the advantages of CMA-ES and sep-CMA-ES in moderate or high dimension, we propose a simple **policy**: using sep-CMA-ES for the first 100 to 200 times n/λ iterations and then CMA-ES retaining all of the acquired strategy parameters.[7] The underlying rationale is that the first 100 to 200 times n/λ iterations are almost negligible compared to the adaptation costs for the full covariance matrix afterwards. In some cases, this policy will be adversarial. Using only CMA-ES will be visibly better if most necessary dependencies can be learned quickly with the CMA-ES in the beginning already. Sticking to sep-CMA-ES will be significantly better if the necessary scaling continuously varies (like on $f^{\beta}_{\mathrm{diffpow}}$) and therefore the fast learning of sep-CMA remains beneficial.

Finally, the sep-CMA will serve as a stepping stone to other variants of CMA with linear time and space complexity that we plan to develop.

References

1. D. Arnold and R. Salomon. Evolutionary gradient search revisited. *IEEE Transactions on Evolutionary Computation*, 11(4):480–495, 2007.
2. 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.
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. 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.
7. J. N. Knight and M. Lunacek. Reducing the space-time complexity of the CMA-ES: Addendum. Errata for [6], `http://www.cs.colostate.edu/~nate/lcmaes/errata.pdf`, March 2008.
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.
10. R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. Research Report 6498, INRIA, April 2008. `http://hal.inria.fr/inria-00270901/en`.

[7] Using sep-CMA can improve the sometimes slow progress of CMA-ES in the very beginning of the optimisation which we ascribe partly to non-elitism and partly to the inability of CMA-ES to quickly reduce variances in single coordinates.