Main Vector Adaptation: A CMA Variant with Linear Time and Space Complexity

Jan Poland
University Tübingen, WSI RA
Sand 1, D - 72076 Tübingen
Germany
poland@informatik.uni-tuebingen.de

Andreas Zell
University Tübingen, WSI RA
Sand 1, D - 72076 Tübingen
Germany
zell@informatik.uni-tuebingen.de

Abstract

The covariance matrix adaptation (CMA) is one of the most powerful self adaptation mechanisms for Evolution Strategies. However, for increasing search space dimension N, the performance declines, since the CMA has space and time complexity $O(N^2)$. Adapting the *main mutation vector* instead of the covariance matrix yields an adaptation mechanism with space and time complexity $O(N)$. Thus, the main vector adaptation (MVA) is appropriate for large-scale problems in particular. Its performance ranges between standard ES and CMA and depends on the test function. If there is one preferred mutation direction, then MVA performes as well as CMA.

1 Introduction

Evolution Strategies need self adaptation, in order to apply to hard or badly scaled fitness functions. For a motivating example, consider the optimization of a prism lens from [4], chaper 9: Given is a glass block that consists of 19 prism segments. The thickness of the segments at both ends is variable. Hence, there are 20 object variables, which are tuned in order to focus the light rays and minimize the overall thickness of the lens (see Fig. 1, the exact fitness function is stated as f_{10} in Section 5). A simple ES with mutative or derandomized step length control finds the focus quite easily, but it fails to minimize the lens thickness. This is due to the fact that all object variables have to be reduced simultaneously in order to minimize the thickness, while any other mutation direction destroys the focus.

Other self adaptation algorithms fail in this situation,

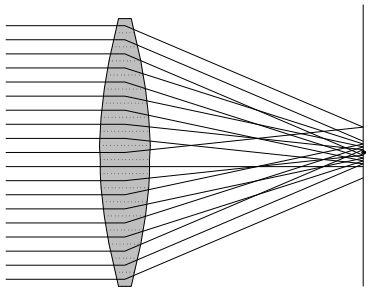

Figure 1: Optimization of a prism lens

too. One could expect for example an adaptation of the mutation mean (momentum adaptation) to be appropriate (compare e.g. [3]). However, our experiments with such an algorithm have not been successful. On the other hand, the covariance matrix adaptation (CMA), which adapts the covariance of the mutation, does work in this situation. In this paper, we will develop a new self adaptation algorithm that also works in this situation and is based on similar ideas as the CMA, but with less space and time consumption.

2 The CMA Algorithm

The covariance matrix adaptation (see [1] or [2]) is one of the most powerful self adaptation mechanisms today available for Evolution Strategies. While a simple ES uses a mutation distributen $N(0, \sigma^2 \cdot I)$ (where I is the identity matrix), the CMA-ES performes a $N(0, \sigma^2 \cdot C)$- distributed mutation, and the covariance matrix C is being adapted. This procedure is based on the following ideas (see [1]): Let $Z_1, \ldots, Z_n$ be independently $N(0,1)$ distributed, $z_1, \ldots, z_n \in \mathbb{R}^N$, $\sigma_1, \ldots, \sigma_n \in \mathbb{R}$ and

$$Z = \sum_{i=1}^{n} Z_i \cdot \sigma_i z_i \quad \text{and} \quad C = \sum_{i=1}^{n} \sigma_i^2 \cdot z_i z_i'.$$

Then Z is a normally distributed random vector with mean 0 and covariance C. On the other hand, any N-dimensional normal distribution $N(0, C)$ can be generated by such a sum by choosing for z_i the eigenvectors and for σ_i^2 the corresponding eigenvalues of C.

Thus, the offspring $\hat{x}$ can be created by adding a $N(0, \sigma^2 \cdot C)$ distributed random vector to the parent x, where $\sigma > 0$ is the global step size for x which can be adapted conventionally. If p_m denotes the mutation path, i.e. the (weighted) mean of the last successful steps, then C can be updated by

$$\hat{C} = (1 - c_{cov}) \cdot C + c_{cov} \cdot \hat{p}_m \hat{p}'_m,$$

where $c_{cov} > 0$ is a small constant and $\hat{C}$ is the covariance matrix for the next generation. The path p_m is updated by a similar formula:

$$\hat{p}_m = (1 - c_m) \cdot p_m + c_m^u \cdot \sigma^{-1}(\hat{x} - x)$$

(note that $\sigma^{-1}(\hat{x} - x)$ is $N(0, C)$ distributed). Again, $c_m > 0$ is a small constant, while $c_m^u = \sqrt{c_m(2 - c_m)}$, which assures that p_m and $\hat{p}_m$ are identically distributed if p_m and $\sigma^{-1}(\hat{x} - x)$ are independent and identically distributed. Hence, the path is not influenced by the global step size σ.

This covariance matrix adaptation procedure has turned out to be very efficient and is successful in cases where the standard ES breaks down. In particular, the CMA makes the strategy invariant against any linear transformation of the search space. Moreover, the covariance matrix approximates the inverse Hessian matrix for functions with sufficient regularity properties. Thus, the CMA can be considered as an evolutionary analogon to quasi Newton optimization algorithms.

The main drawback of the CMA comes with increasing dimension N of the search space. The storage space and the update time for the covariance matrix have complexity $O(N^2)$, while the computation of the eigenvectors and eigenvalues is even $O(N^3)$. This can be reduced to $O(N^2)$ by executing the step for example after $N/10$ generations instead of every generation, which does no severe damage. In any case, for large N, the CMA performance declines rapidly, compare also Fig. 7. There are other self adaptation mechanisms which are similar to CMA, such as the rotation angle adaptation (see [5]). This algorithm has quadratic space and time complexity as well and shows a poorer performance in general.

3 Main Vector Adaptation

For many functions the advantage of the CMA compared to a conventional ES is given by the fact that the CMA finds the *preferred mutation direction*, while all other directions are not acceptable. An instance is the lens optimization (see the introduction). In these cases, it should be sufficient to adapt one vector instead of an entire matrix in order to find this direction. This is done basically by the simple formula $\hat{v} = (1 - c_v) \cdot v + c_v \cdot \hat{p}_m$, where p_m is the path as before and $c_v > 0$ is a small constant. Then, the offspring $\hat{x}$ can be generated by $\hat{x} = x + \sigma \cdot Z + \sigma \cdot Z_1 \cdot v$, where $Z \sim N(0, I)$ and $Z_1 \sim N(0, 1)$. We call v the *main (mutation) vector* and the algorithm *main vector adaptation* (MVA).

In order to make these formulas work in practice, we have to regard two details. First, the mutation is independent of the sign of the main vector v. However, in contrast to the update formula for the covariance matrix, the update of v depends on the sign of the path p_m. This can result in the annihilation of subsequent mutation steps and inhibits the adaptation of v, in particular for difficult functions such as the sharp ridge f_7 (cf. Section 5). To avoid this breakdown, we simply flip v if necessary:

$$\hat{v} = (1 - c_v) \cdot \text{sign}(\langle v, p_m \rangle) \cdot v + c_v \cdot \hat{p}_m.$$

Here, $\langle \cdot, \cdot \rangle$ denotes the scalar product of two vectors.

The second problem to be fixed is the standard deviation of $Z + Z_1 \cdot v$ along the main vector v. Since $Z + Z_1 \cdot v \sim N(0, I + vv')$, its variance along v is $\sigma_v^2 = 1 + \|v\|^2$, hence $\sigma_v = \sqrt{1 + \|v\|^2}$. On the contrary, $\sigma_v = 1 + \|v\|$ would be desired, since this corresponds to the functioning of v as additional mutation in the main vector direction. Thus, we write

$$\hat{x} = x + \sigma \cdot (Z + Z_1 \cdot w_v \cdot v).$$

Letting $w_v = 1 + 2 \cdot \|v\|^{-1}$ yields $\sigma_v = 1 + \|v\|$, however, the experiments show that a constant $w_v = 3$ (corresponding to $\|v\| = 1$) yields the best results in general, occasionally, $w_v = 1$ is better.

Again, we point out that taking v (or anything similar) as the *mean vector* of the mutation does not yield an efficient algorithm! This is presumably due to the geometry of the high dimensional $\mathbb{R}^N$, where a nonzero mutation mean results in a shifted sphere, while the additional main vector mutation yields an ellipsoid as mutation shape.

4 The MVA-ES Algorithm

In order to adapt the mutation step size σ, we employ the same derandomized mechanism using a path p_σ, with the only difference that for p_σ the main vector v

is ignored. This is again a perfect analogy to the CMA ([1]). Thus, the complete mutation algorithm reads as follows.

Mutation.

1. Generate $Z \sim N(0, I)$
2. Generate $Z_1 \sim N(0, 1)$
3. $\hat{x} = x + \sigma \cdot (Z + Z_1 \cdot w_v \cdot v)$
4. $\hat{p_\sigma} = (1 - c_\sigma) \cdot p_\sigma + c_\sigma^u \cdot Z$
5. $\hat{\sigma} = \sigma \cdot \exp\left((\|\hat{p_\sigma}\| - \hat{\chi}_N)/(d_\sigma \cdot \hat{\chi}_N) \right)$
6. $\hat{p}_m = (1 - c_m) \cdot p_m + c_m^u \cdot (Z + Z_1 \cdot w_v \cdot v)$
7. $\hat{v} = (1 - c_v) \cdot \text{sign}(\langle v, p_m \rangle) \cdot v + c_v \cdot \hat{p_m}$

where

$Z \in \mathbb{R}^N$ and $Z_1 \in \mathbb{R}$ random vectors,
$x, \hat{x} \in \mathbb{R}^N$ parent and offspring individuals,
$p_\sigma, \hat{p}_\sigma \in \mathbb{R}^N$ parent and offspring σ-paths,
$c_\sigma > 0$ σ-path constant and $c_\sigma^u = \sqrt{c_\sigma(2 - c_\sigma)}$, choose e.g. $c_\sigma = 4/(N + 4)$,
$\sigma, \hat{\sigma} > 0$ parent and offspring mutation step lengths,
$p_m, \hat{p}_m \in \mathbb{R}^N$ parent and offspring paths,
$c_m > 0$ path constant and $c_m^u = \sqrt{c_m(2 - c_m)}$, choose e.g. $c_m = 4/(N + 4)$,
$v, \hat{v} \in \mathbb{R}^N$ parent and offspring main vectors,
$c_v > 0$ main vector constant, choose e.g. $c_v = 2/(N + \sqrt{2})^2$,
$\hat{\chi}_N = \text{E}(\|N(0, I)\|) = \sqrt{2} \cdot \Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2}) \approx \sqrt{N - \frac{1}{2}}$ (we prefer this approximation to the approximation from [1]).

The suggestions for the parameters c_σ, c_m and c_v are the same as the respective suggestions for the CMA in [1], they are good also for MVA. However, for some test functions, a greater value c_v yields a faster convergence, e.g. $c_v = 0.1$.

For the recombination, we restrict here to a simple intermediate recombination that is executed by computing the mean of the object variables, paths, step sizes, and main vectors of all participating individuals. Clearly, other recombination types are possible as well, e.g. discrete or generalized intermediate recombination.

5 Experimental Results

The MVA-ES has been tested against the CMA-ES and a standard ES with derandomized step length control. We used the test functions $f_1, \ldots, f_9$ from [1], which are nonlinear and resistent to simple hill-climbing. In addition, we test the lens optimization f_{10}. In order to obtain non-separability for $f_1, \ldots, f_9$, one determines a random orthonormal basis U before every ES run and minimizes $f_k(U \cdot x)$ instead of $f_k(x)$. Comparing the results to the case $U = I$ shows that each of the tested algorithms is invariant against any rotation of the search space. This was of course expected.

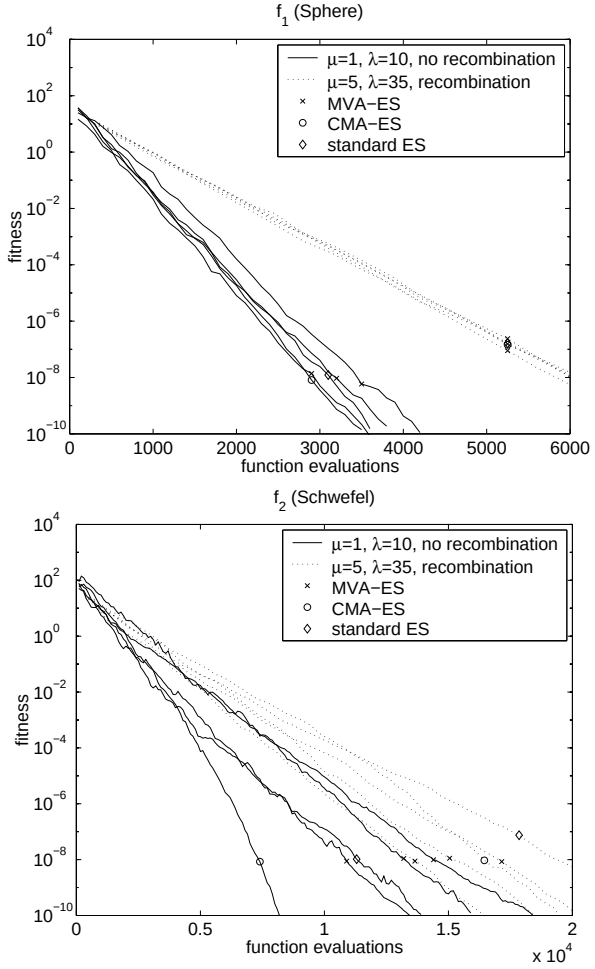

Figure 2: f_1 (sphere) and f_2 (Schwefel)

In order to compare the adapation properties, all functions have been tested in dimension $N = 20$. Moreover, we obtain a time and space complexitiy comparison with function f_1 in different dimensions. For each test function and each ES, we try a simple $(1, 10)$ variant without recombination (solid line in the plots) and a (5,35) variant with intermediate recombination (dotted line). We performed 70 runs for each setting.

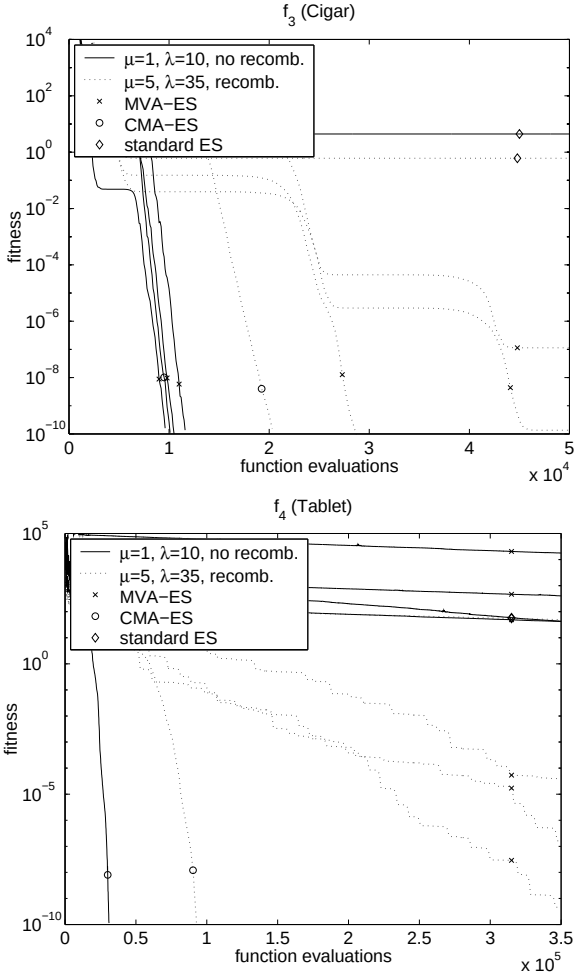

Figure 3: f_3 (cigar) and f_4 (tablet)

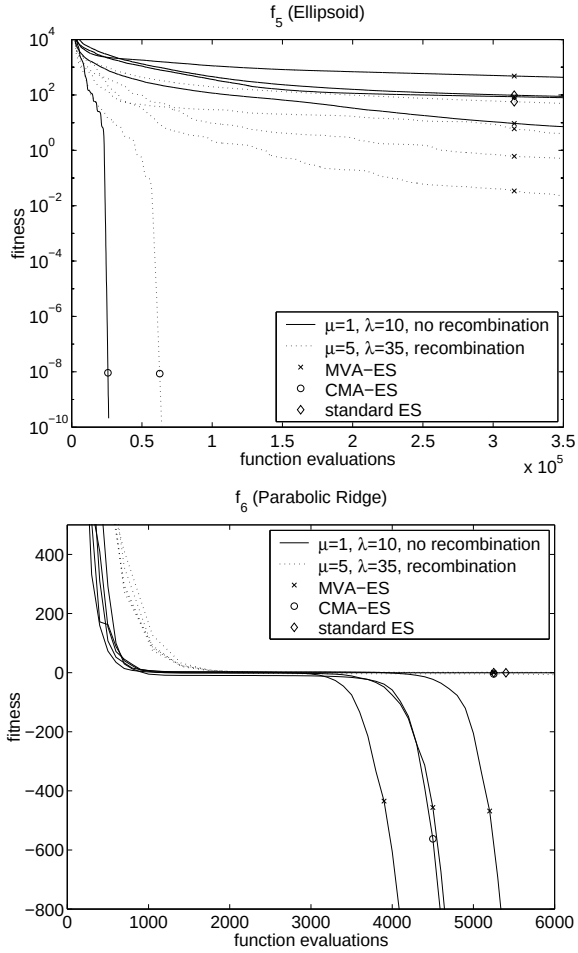

Figure 4: f_5 (ellipsoid) and f_6 (parabolic ridge)

The plots show the fitness curves of average runs of MVA-ES, CMA-ES and standard ES. For MVA-ES, the best and the worst fitness curves are displayed, too. The tests have been performed with MATLAB, for the CMA-ES we used the implementation from [1].

Function $f_1(x) = \sum_{i=1}^{N} x_i^2$ is the sphere function and the only one that remains separable under the random rotation. We observe that neither ES has difficulties to find the optimum, as well as for Schwefels function $f_2(x) = \sum_{i=1}^{N}(\sum_{j=1}^{i} x_j^2)$. The "cigar" $f_3(x) = x_1^2 + \sum_{i=2}^{N}(1000x_i)^2$ is more interesting: The standard ES fails, while CMA and MVA are successful. This is the classical case of one preferred mutation direction: It is easy to optimize the coordinates $2 \dots N$, but then the remaining feasible direction along the first coordinate is difficult to find. We observe further that recombination apparantly disturbs the main vector adaptation a little in general.

The "tablet" $f_4(x) = (1000x_1)^2 + \sum_{i=2}^{N} x_i^2$ is in a sense the converse of f_3: The optimization is first carried out along the first coordinate, then the coordinates $2 \dots N$ remain. Since there is no preferred mutation direction, it is not unexpected that MVA fails to converge, while the covariance matrix adapts easily to this situation. Here a mechanism that "fades out" one direction, i.e. the inverse of MVA could be suitable. Note that recombination helps the MVA in this case to converge. The ellipsoid $f_5(x) = \sum_{i=1}^{N}(1000^{\frac{i-1}{N-i}} x_i)^2$ is another linear transformation of the sphere. Here, there is neither a preferred mutation direction as in f_3 nor an "anti-mutation" direction as in f_4. Again, CMA adapts easily, while the bad scaling remains a problem for MVA and standard ES.

The parabolic ridge $f_6(x) = -x_1 + 100\sum_{i=2}^{N} x_i^2$ is an instance for a preferred mutation direction and is easily optimized by the MVA-ES. The same is true for the

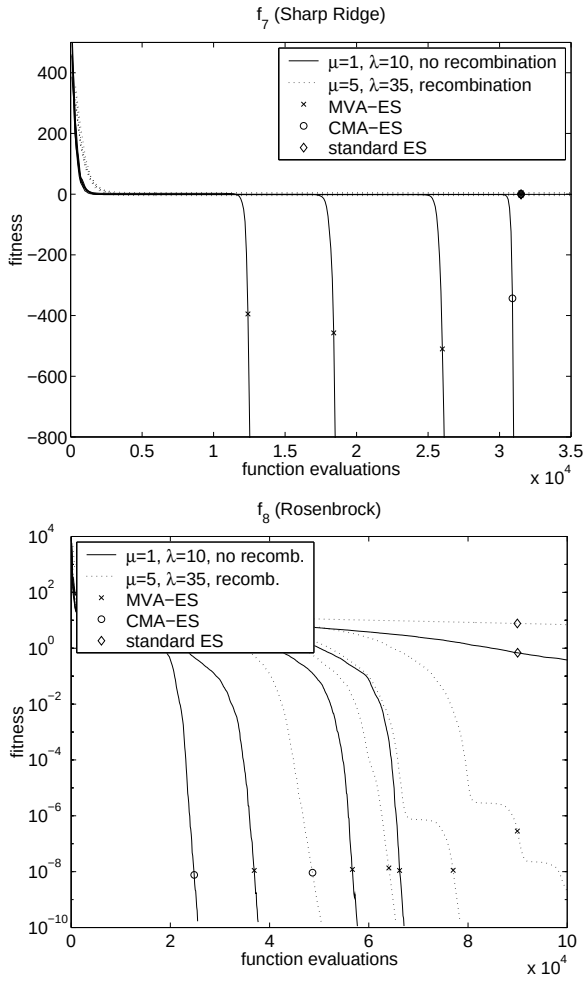

Figure 5: f_7 (sharp ridge) and f_8 (Rosenbrock)

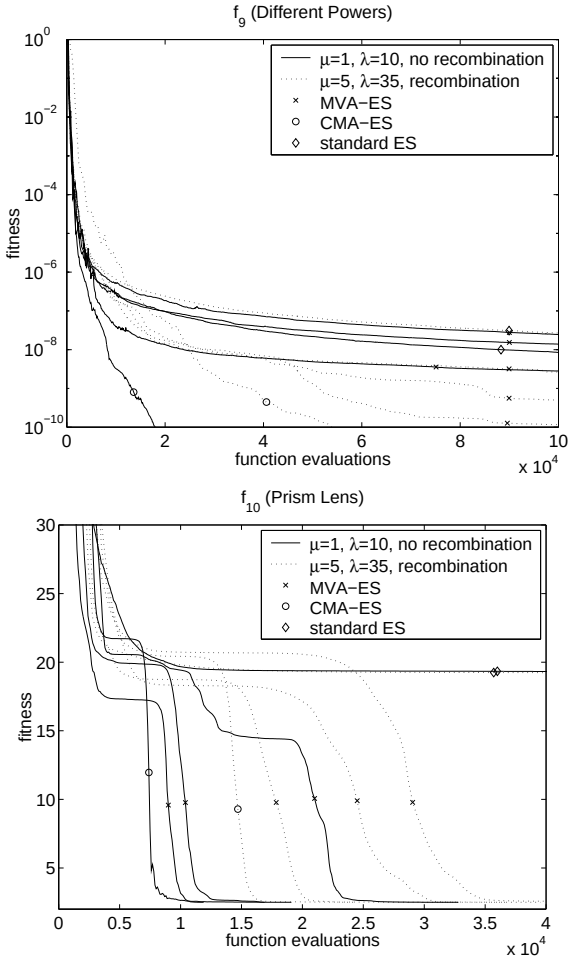

Figure 6: f_9 (different powers) and f_{10} (prism lens)

sharp ridge $f_7(x) = -x_1 + 100\sqrt{\sum_{i=2}^{N} x_i^2}$. This function is particularly hard to optimize, since the local gradient is constant. A too small choice for w_v results in a failure of the MVA. The generalized Rosenbrock function $f_8(x) = \sum_{i=1}^{N-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$ ("banana function") is an instance for a bent ridge. Again, MVA and CMA are successful in this situation, while standard ES fails. On the contrary, function $f_9(x) = \sum_{i=1}^{N} |x_i|^{2+10\frac{i-1}{N-1}}$, a sum of different powers, is hard for MVA. Recombination improves the MVA convergence.

Function $f_{10}(x) = \sum_{i=1}^{N-1} \left(R - \frac{h}{2} - h \cdot (i-1) - \frac{b}{h} \cdot (\epsilon - 1)(x_{i+1} - x_i)\right)^2 + \max_i x_i + \min_i x_i$ is the prism lens function (see introduction and Fig. 1). Here, $h > 0$ is the height of the segments, $b > 0$ is the distance from the lens to the screen, $R = h \cdot \frac{N-1}{2}$ the y-coordinate of the desired focal point, $\epsilon > 1$ the refraction index of the lens and $2 \cdot x_i$ the respective thickness. While the standard ES stagnates with a thickness induced by the initial random initialization, MVA and CMA find the optimum. When the preferred direction has been found and the lens is thinned, the focus is slightly disturbed and has to be restored afterwards. The MVA-ES does this much more rapidly than the CMA-ES, when $c_v = 0.1$ is chosen. (The corresponding parameter setting for CMA does not work.)

Finally, we compare the time and space consumption of a $(1, 10)$-MVA-ES and a $(1, 10)$-CMA-ES in search space dimension $N = 2, 5, 10, 20, 50, 100, 200, 400, 800$, see Fig. 7. In the time complexity plot, the average time for one generation with test function f_1 is displayed. The covariance matrix eigenvectors have been updated all $N/10$ generations. The performance decrease is linear in N for the MVA and quadratic in N for the CMA. Of course, the time complexity argu-

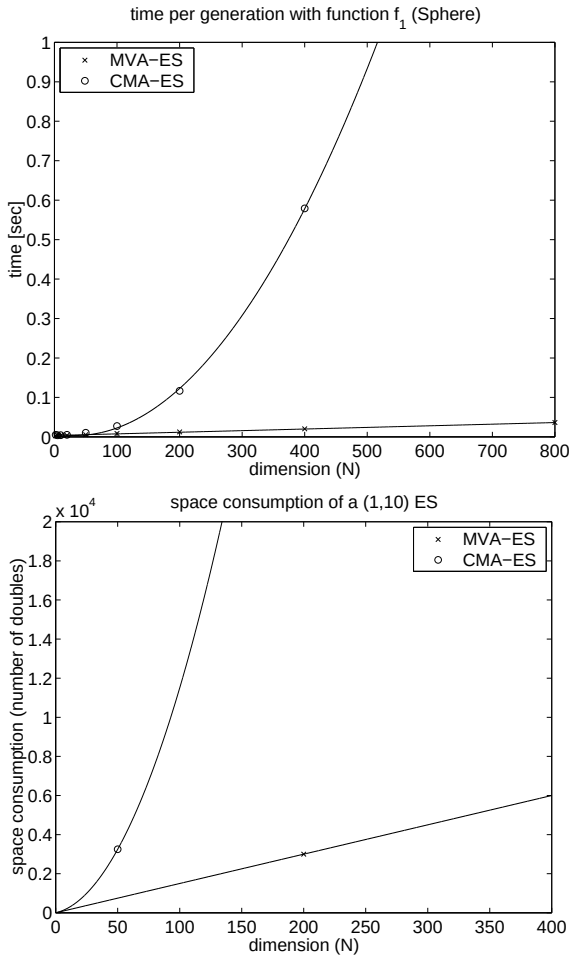

Figure 7: Time and space consumption of the $(1, 10)$-MVA-ES and the $(1, 10)$-CMA-ES for increasing dimension N. The time for one generation with test function f_1 is shown, the covariance matrix eigenvectors have been updated all $N/10$ generations.

ment becomes unimportant when the fitness function is expensive, in particular when its time complexity in N is greater or equal $O(N^2)$. But even then, the space advantage of the MVA remains, especially if the population consists of many individuals. Moreover, for large N, the computation time for the covariance matrix eigenvectors increases drastically in practice, since there is only a limited amount of memory available. For a Pentium III with 128 MB RAM and the built-in MATLAB function, this occurs at about $N \approx 400$.

6 Conclusions

We presented a new self adaptation mechanism for Evolution Strategies that adapts the main mutation vector. The algorithm is similar to CMA and shows a similar performance in situations where there is one preferred mutation direction to find. If the demanded adaptation is more complex, MVA is less powerful than CMA. On the other hand, the time and space complexity of MVA is only linear in the search space dimension N. Therefore, MVA is appropriate for problems in high dimensional search spaces ($N > 500$), where the use of CMA becomes problematic because of its $O(N^2)$ complexity. In low dimensions ($N < 100$), CMA will remain the better choice.

There are several possible extensions of MVA. For example, one could adapt an "anti-mutation" vector, this can be appropriate for functions similar to f_4. One can adapt more than one main vector, controlled e.g. by the scalar product. If this is extended to N vectors, it could be possible to obtain an algorithm similar to CMA that does not need any eigenvector decomposition and thus has an $O(N^2)$ update of the mutation directions instead of $O(N^3)$.

Acknowledgments.

We would like to thank Nikolaus Hansen for providing us with the CMA-ES code. Furthermore, we thank Jürgen Wakunda and Kosmas Knödler for helpful discussions. This research has been supported by the BMBF (grant no. 01 IB 805 A/1).

References

[1] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. Preprint 2000, to appear in: Evolutionary Computation, Special Issue on Self-Adaptation.

[2] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i, \lambda)$-cma-es. In *5th European Congress on Intelligent Techniques and Soft Computing*, pages 650–654, 1997.

[3] A. Ostermeier. An evolution strategy with momentum adaptation of the random number distribution. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 197–206, 1992.

[4] I. Rechenberg. *Evolutionsstrategie '94*. frommann-holzboog, Stuttgart, 1994.

[5] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, New York, 1995.