Design Principles for Matrix Adaptation Evolution Strategies

Hans-Georg Beyer
Hans-Georg.Beyer@fhv.at
https://homepages.fhv.at/hgb/

Research Center Business Informatics
Vorarlberg University of Applied Sciences

FHV

1. A Short Recap of Evolution Strategies
2. The Matrix Adaptation Idea
3. How to Get the Most Out of It
4. Design Principles for MA-ES on Constrained Problems
5. Summary
6. Related Publications

A Short Recap of Evolution Strategies[1]

Evolution Strategies (ESs) are a class of Evolutionary Algorithms that use:

1. mutation and recombination to generate λ offspring from μ parents
2. perform truncation (aka breeding) selection denoted by

(μ, λ)**:** only the μ best offspring individuals are selected as parents of the next generation, or

$(\mu + \lambda)$**:** both the λ offspring and the μ parent are together are object of selection to determine the parents of the next generation

Remark:

Application domains (search spaces):

- discrete, combinatorial, real-valued optimization and mixtures
- however, predominantly used in real-valued un-constrained optimization
- popularized by the Covariance Matrix Adaptation (CMA) ES[2]

[1] See H.-G. Beyer: Scholarpedia: *Evolution Strategies.* And H.-G. Beyer & H.-P. Schwefel: *Evolution Strategies: A Comprehensive Introduction.* Natural Computing 1(1):3–52, 2002.

[2] See N. Hansen, S.D. Müller, and P. Koumoutsakos. *Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES).* Evolutionary Computation, 11(1):1–18, 2003.

- compared to other EAs, especially Differential Evolution (DE), there is relatively little follow-up work that builts on the CMA-ES
- What are the reasons?
- at first glance CMA-ES seems a rather sophisticated EA
- CMA-ES contains ingredients that seems to be difficult to modify without sacrificing its superb performance on certain test function sets
- most modifications proposed are rather minor and are built around the covariance matrix :
- the tenet is to estimate the covariance matrix
- Why do we need the covariance matrix?
- actually, one wants to mutate parents to get promising offspring
- one only has to generate correlated mutations in order to target into promising directions in the search space
- How to do that without covarianc matrix calculations will be a topic of this tutorial

But, first let us consider a simple ES without correlated mutations:

Simple ES with Self-Adaptation and Recombination

$(\mu/\mu_I, \lambda)$-σSA-ES	line
Initialize $(\mathbf{x}, \sigma, \tau)$	1
Repeat	2
For $l := 1$ **To** λ	3
$\tilde{\sigma}_l := \sigma e^{\tau \mathcal{N}_l(0,1)}$	4
$\tilde{\mathbf{d}}_l := \boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I})$	5
$\tilde{\mathbf{x}}_l := \mathbf{x} + \tilde{\sigma}_l \tilde{\mathbf{d}}_l$	6
$\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$	7
$\tilde{\mathfrak{a}}_l := (\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\sigma}_l)$	8
End	9
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$	10
$\mathbf{x} := \langle \tilde{\mathbf{x}} \rangle$	11
$\sigma := \langle \tilde{\sigma} \rangle$	12
Until(Termination_Condition)	13
Return$(\mathbf{x})$	14

- **Task:** optimize $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^N$ (i.e. unconstrained)
- L3–8: produce λ offspring
- L4: mutate σ (mutation strength), $\tau = 1/\sqrt{2N}$
- L5: generate search direction
- L6: mutate parent by $\mathbf{w} = \sigma\tilde{\mathbf{d}}$
- L7: evaluate offspring
- L8: assemble offspring
- L11f: recombine the *best* μ offspring' $\mathbf{x}$ and σ:[3]

$$\langle \tilde{\mathbf{x}} \rangle := \frac{1}{\mu} \sum_{m=1}^{\mu} \mathbf{x}_{m;\lambda} \quad (1)$$

$$\langle \tilde{\sigma} \rangle := \frac{1}{\mu} \sum_{m=1}^{\mu} \sigma_{m;\lambda} \quad (2)$$

[3] "$m;\lambda$" is the index of the mth best individual out of λ offspring (w.r.t. fitness).

1. Isotropic Gaussian Mutations in $\mathbb{R}^N$

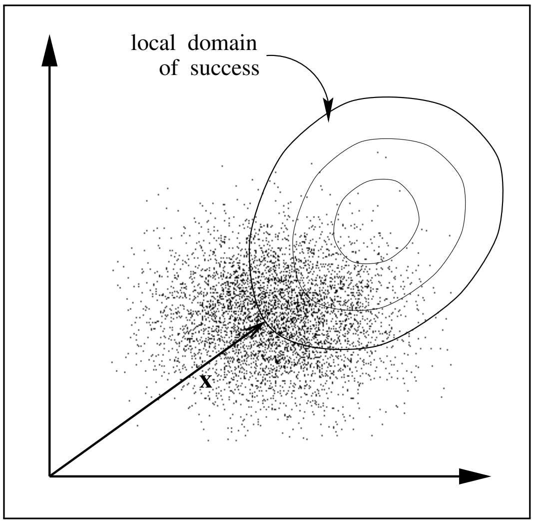


Figure 1: Isotropic Gaussian mutation samples in a 2-dimensional search space applied to a recombinant state $\mathbf{x} = \langle \tilde{\mathbf{x}} \rangle$ in Line 6, Slide 5.

- for "well shaped" local success domains, isotropic Gaussian mutations $\mathbf{w} = \sigma\tilde{\mathbf{d}}$ are sufficient:

$$\mathbf{w} \sim \left(\mathcal{N}(0,\sigma^2), \ldots, \mathcal{N}(0,\sigma^2)\right)^{\mathrm{T}} = \sigma \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}) \quad (3)$$

- probability density function:

$$p(\mathbf{w}) = p(w_1, \ldots, w_N) = \frac{1}{(\sqrt{2\pi}\sigma)^N} \exp\left(-\frac{z_1^2 + \ldots + z_N^2}{2\sigma^2}\right) \quad (4)$$

- (hyper) surfaces of constant p are spherical shells
- note, in high N-dimensional spaces the mutation vectors $\mathbf{w}$ are nearly located in the vicinity of a sphere of radius $\sigma\sqrt{N}$
- mutation strength σ can be adapted by, e.g., $(\mu/\mu_I, \lambda)$-σSA-ES, Slide 5

2. Non-correlated independently distributed Gaussian mutations

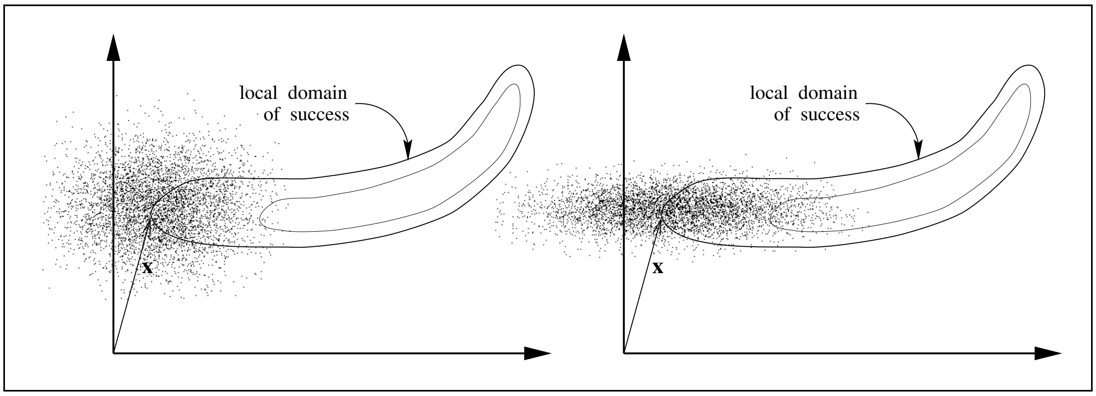


Figure 2: Success domains with preference directions parallel to certain coordinate directions are better treated by Gaussian mutation vectors the components of which have different mutations strengths (lhs: isotropic, rhs: ellipsoidal mutations).

$$\mathbf{w} \sim \left(\mathcal{N}(0,\sigma_1^2),\ldots,\mathcal{N}(0,\sigma_N^2)\right)^{\mathrm{T}}, \quad p(\mathbf{w}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{w_i^2}{2\sigma_i^2}\right) \tag{5}$$

- there is a set of N strategy parameters σ_i to be evolved

$$(\sigma_1,\ldots,\sigma_i,\ldots,\sigma_N)^{\mathrm{T}} \tag{6}$$

- (hyper) surfaces of constant p are axes-parallel ellipsoids

3. Correlated Gaussian mutations

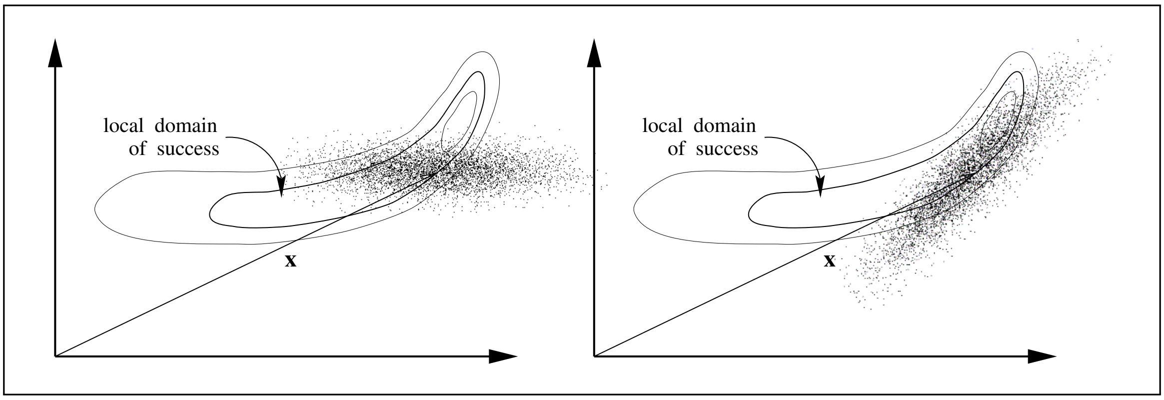


Figure 3: *Rotated* mutation ellipsoids (rhs) are better suited for the recombinant **x**.

- *correlated* **w** mutations are to be used to obtain mutation ellipsoids arbitrarily oriented in search space

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0},\mathbf{\Sigma}) \tag{7}$$

$$p(\mathbf{w}) = \frac{1}{\left(\sqrt{2\pi}\right)^N} \frac{1}{\sqrt{\det[\mathbf{\Sigma}]}} \exp\left(-\frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{w}\right) \tag{8}$$

- $\mathbf{\Sigma}$ is the *covariance matrix* and $\mathbf{\Sigma}^{-1}$ its inverse

Remarks

- covariance matrix $\mathbf{\Sigma}$ contains $N(N+1)/2$ independent parameters to be learned (because $\mathbf{\Sigma}$ is symmetric)
- ➥ there are N object parameters to be evolved in order to optimize $f(\mathbf{x})$, *but,* there are $N(N+1)/2$ $\mathbf{\Sigma}$-matrix components to be learned, too!
- ➥ using an EA that learns correlated mutations via covariance matrix $\mathbf{\Sigma}$ makes sense only when one has a budget of function evaluations that is greater than kN^2

How to generate correlated mutations?

- correlated mutations $\mathbf{w}$ can be produced by linear transformation of iid standard normally distributed vectors $\mathbf{z} = (\mathcal{N}_1(0,1), \ldots, \mathcal{N}_N(0,1))^{\mathrm{T}}$ by a two-step process
  1. calculating the direction: $\mathbf{d} := \mathbf{Mz}$
  2. scaling the length: $\mathbf{w} := \sigma\mathbf{d}$
- since $\mathrm{E}[\mathbf{w}] = \mathrm{E}[\sigma\mathbf{Mz}] = \sigma\mathbf{M}\mathrm{E}[\mathbf{z}] = \mathbf{0}$, one finds using the definition of $\mathbf{\Sigma}$

$$\mathbf{\Sigma} = \mathrm{E}[\mathbf{ww}^{\mathrm{T}}] = \sigma^2\mathrm{E}[\mathbf{Mzz}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}] = \sigma^2\mathbf{M}\mathrm{E}[\mathbf{zz}^{\mathrm{T}}]\mathbf{M}^{\mathrm{T}} = \sigma^2\mathbf{MM}^{\mathrm{T}} \qquad (9)$$

The Matrix Adaptation Idea

How can one get M if Σ were known?

☞ take the "square root" of $\mathbf{\Sigma}$ in (9), i.e., $\mathbf{M} = \frac{1}{\sigma}\sqrt{\mathbf{\Sigma}}$
This can be done by:

- CHOLESKY-decomposition
- matrix square root via eigenvalue decomposition

However, how can one get the covariance matrix Σ?

- it must be derived from the evolutionary dynamics of the real ES run
- this is what the Covariance Matrix Adaptation (CMA) ES does

But, why not deriving the M matrix from the observed ES dynamics directly?

- analysis of the CMA-ES revealed that one can (approximately) rewrite the CMA-ES algorithm and remove its "C"
- as a result one obtains very simple Matrix Adaptation (MA) ESs that perform equally well as the CMA-ES[4]

☞ **C** related numerical operations are no longer needed!

[4] H.-G. Beyer and B. Sendhoff. Simplify Your Covariance Matrix Adaptation Evolution Strategy. *IEEE Transactions on Evolutionary Computation* 21(5):746–759, 2017. DOI: 10.1109/TEVC.2017.2680320

A Simple Recombinative MA-ES with Self-Adaptation (SA)

$(\mu/\mu_I,\lambda)$-σSA-MA-ES	line
Initialize $(\mathbf{x},\sigma,\tau,\tau_{\mathrm{M}},\mathbf{M}:=\mathbf{I})$	1
Repeat	2
For $l:=1$ **To** λ	3
$\tilde{\sigma}_l:=\sigma e^{\tau\mathcal{N}_l(0,1)}$	4
$\tilde{\mathbf{z}}_l:=\mathcal{N}_l(\mathbf{0},\mathbf{I})$	5
$\tilde{\mathbf{d}}_l:=\mathbf{M}\tilde{\mathbf{z}}_l$	6
$\tilde{\mathbf{x}}_l:=\mathbf{x}+\tilde{\sigma}_l\tilde{\mathbf{d}}_l$	7
$\tilde{f}_l:=f(\tilde{\mathbf{x}}_l)$	8
$\tilde{\mathfrak{a}}_l:=\left(\tilde{f}_l,\tilde{\mathbf{x}}_l,\tilde{\sigma}_l,\tilde{\mathbf{z}}_l\right)$	9
End	10
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1,\ldots,\tilde{\mathfrak{a}}_\lambda)$	11
$\mathbf{x}:=\langle\tilde{\mathbf{x}}\rangle$	12
$\sigma:=\langle\tilde{\sigma}\rangle$	13
$\mathbf{M}:=\mathbf{M}\left[\mathbf{I}+\frac{1}{\tau_{\mathrm{M}}}\left(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle-\mathbf{I}\right)\right]$	14
Until(Termination_Condition)	15

- L3–9: produce λ offspring
- L4: mutate σ (mutation strength), $\tau:=1/\sqrt{2N}$
- L5f: generate search direction
- L7: mutate parent by $\mathbf{w}=\tilde{\sigma}\tilde{\mathbf{d}}$
- L8: evaluate offspring
- L9: assemble offspring
- L12f: recombine the *best* μ offspring' $\mathbf{x}$ and σ, (1/2)
- L14: update $\mathbf{M}$-matrix with learning rate

$$\tau_{\mathrm{M}}:=2+\frac{(N+1)N}{\mu} \tag{10}$$

$$\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle:=\frac{1}{\mu}\sum_{m=1}^{\mu}\tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^{\mathrm{T}} \tag{11}$$

Comparison to Covariance Matrix Self-Adaptation ES (CMSA-ES) [5]

$(\mu/\mu_I,\lambda)$-σ-CMSA-ES	line
Initialize $(\mathbf{x},\sigma,\tau,\tau_{\mathrm{c}},\mathbf{C}:=\mathbf{I})$	1
Repeat	2
For $l:=1$ **To** λ	3
$\tilde{\sigma}_l:=\sigma e^{\tau\mathcal{N}_l(0,1)}$	4
$\tilde{\mathbf{d}}_l:=\sqrt{\mathbf{C}}\,\mathcal{N}_l(\mathbf{0},\mathbf{I})$	5
$\tilde{\mathbf{x}}_l:=\mathbf{x}+\tilde{\sigma}_l\tilde{\mathbf{d}}_l$	6
$\tilde{f}_l:=f(\tilde{\mathbf{x}}_l)$	7
$\tilde{\mathfrak{a}}_l:=\left(\tilde{f}_l,\tilde{\mathbf{x}}_l,\tilde{\sigma}_l,\tilde{\mathbf{d}}_l\right)$	8
End	9
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1,\ldots,\tilde{\mathfrak{a}}_\lambda)$	10
$\mathbf{x}:=\langle\tilde{\mathbf{x}}\rangle$	11
$\sigma:=\langle\tilde{\sigma}\rangle$	12
$\mathbf{C}:=\left(1-\frac{1}{\tau_{\mathrm{c}}}\right)\mathbf{C}+\frac{1}{\tau_{\mathrm{c}}}\langle\tilde{\mathbf{d}}\tilde{\mathbf{d}}^{\mathrm{T}}\rangle$	13
Until(Termination_Condition)	14

Differenzes to $(\mu/\mu_I,\lambda)$-σSA-MA-ES:

- L5: generate correlated search direction, matrix $\sqrt{\mathbf{C}}$ must be calculated in an $O(N^3)$ step
- L13: update $\mathbf{C}$-matrix with learning rate:

$$\tau_{\mathrm{c}}:=1+\frac{(N+1)N}{2\mu} \tag{12}$$

$$\langle\tilde{\mathbf{d}}\tilde{\mathbf{d}}^{\mathrm{T}}\rangle:=\frac{1}{\mu}\sum_{m=1}^{\mu}\tilde{\mathbf{d}}_{m;\lambda}\tilde{\mathbf{d}}_{m;\lambda}^{\mathrm{T}} \tag{13}$$

[5] H.-G. Beyer and B. Sendhoff, *Covariance Matrix Adaptation Revisited – the CMSA Evolution Strategy*, in PPSN X, pp. 123–132, Berlin: Springer, 2008.

Understanding the M-Update

- What do the $\mathbf{z}$ variations see in MA-ES algorithm, Line 5, Slide 11?

$$f(\tilde{\mathbf{x}}) = f(\mathbf{x} + \sigma \mathbf{M}\mathbf{z}) =: g(\mathbf{z}) \tag{14}$$

- assume $g(\mathbf{z})$ defines quadratic fitness landscapes

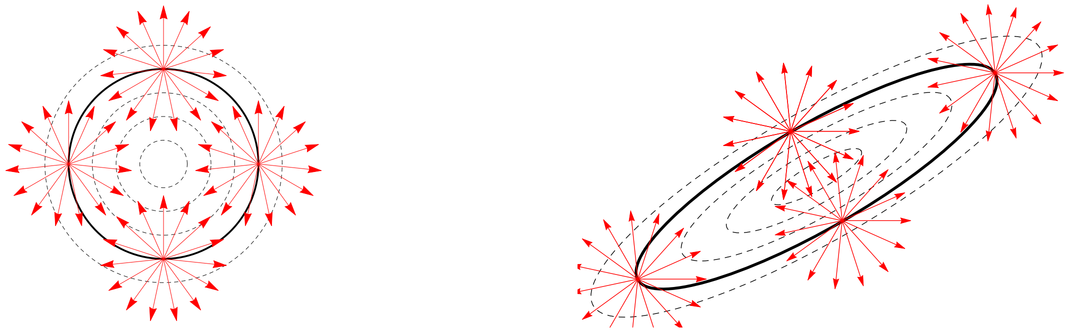


Figure 4: Isotropy in search space considered from "viewpoint" of the $\mathbf{z}$ variations in Line 5, Slide 11: The black curves represent lines of constant f-values. The isotropic $\mathbf{z} \sim \mathcal{N}_l(\mathbf{0}, \mathbf{I})$ vectors experience on average the same selective pressure in case of the spherical success domain (left graph) independent of the location of parental state $\mathbf{x}$. Thus, there are no correlations in the $\mathbf{z}$-vectors implying $\mathrm{E}[\langle \mathbf{z}\mathbf{z}^{\mathrm{T}} \rangle] \propto \mathbf{I}$. In the case of an elliptical success domain (right graph) symmetry is broken and the $\mathbf{z}$ experience different selective pressure in different directions. This implies $\mathrm{E}[\langle \mathbf{z}\mathbf{z}^{\mathrm{T}} \rangle] \not\propto \mathbf{I}$.

- $\mathbf{M}$-update in Line 14, Slide 11:

$$\mathbf{M} := \mathbf{M}\left[\mathbf{I} + \tfrac{1}{\tau_{\mathrm{M}}}\left(\langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}} \rangle - \mathbf{I}\right)\right] \tag{15}$$

☞ change of $\mathbf{M}$ is governed by the deviation of the $\langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}} \rangle$-matrix from the identity matrix $\mathbf{I}$

- taking the expectation

$$\mathrm{E}[\mathbf{M}] = \mathbf{M}\left[\mathbf{I} + \tfrac{1}{\tau_{\mathrm{M}}}\left(\mathrm{E}[\langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}} \rangle] - \mathbf{I}\right)\right] \tag{16}$$

- if $\mathrm{E}[\langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}} \rangle] = \alpha \mathbf{I} \implies$ $\mathbf{M}$ is only changed by a scalar factor

☞ the $\mathbf{z}$-vectors "see" a sphere

- if $\mathrm{E}[\langle \tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}} \rangle] \neq \alpha \mathbf{I} \implies$ $\mathbf{z}$-vectors "experience" an anisotropic fitness landscape

☞ $\mathbf{M}$ undergoes changes during evolution

☞ a general quadratic fitness landscape (ellipsoidally shaped) is gradually transformed into a spherical landscape

Reducing the Internal Costs of the MA-ES – The Fast MA-ES

- most expensive Line 14: $\mathcal{O}\left(\mu N^3\right)$, N – search space dimensionality
- costs for generating a single offspring: $\mathcal{O}\left(\frac{\mu N^3}{\lambda}\right) = \mathcal{O}\left(N^3\right)$
- however, this is the naive view
- recasting Line 14, Slide 11

$$\begin{aligned}
\mathbf{M} &:= \mathbf{M}\left[\mathbf{I} + \frac{1}{\tau_\mathrm{M}}\left(\langle \mathbf{z}\mathbf{z}^\mathrm{T}\rangle - \mathbf{I}\right)\right] \\
&= \left(1 - \frac{1}{\tau_\mathrm{M}}\right)\mathbf{M} + \mathbf{M}\frac{1}{\tau_\mathrm{M}}\langle \mathbf{z}\mathbf{z}^\mathrm{T}\rangle \\
&= \left(1 - \frac{1}{\tau_\mathrm{M}}\right)\mathbf{M} + \frac{1}{\tau_\mathrm{M}}\langle (\mathbf{M}\mathbf{z})\mathbf{z}^\mathrm{T}\rangle \\
&\overset{(\mathrm{L6})}{=} \left(1 - \frac{1}{\tau_\mathrm{M}}\right)\mathbf{M} + \frac{1}{\tau_\mathrm{M}}\langle \mathbf{d}\mathbf{z}^\mathrm{T}\rangle
\end{aligned} \tag{17}$$

$\Rightarrow$ costs are effectively reduced to: $\mathcal{O}\left(\frac{\mu N^2}{\lambda}\right) = \mathcal{O}\left(N^2\right)$

Simple MA-ES with Self-Adaptation (SA) – Fast Version

$(\mu/\mu_I, \lambda)$-σSA-fMA-ES	line
Initialize $(\mathbf{x}, \sigma, \tau, \tau_\mathrm{M}, \mathbf{M} := \mathbf{I})$	1
Repeat	2
For $l := 1$ **To** λ	3
$\tilde{\sigma}_l := \sigma \mathrm{e}^{\tau \mathcal{N}_l(0,1)}$	4
$\tilde{\mathbf{z}}_l := \boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I})$	5
$\tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l$	6
$\tilde{\mathbf{x}}_l := \mathbf{x} + \tilde{\sigma}_l \tilde{\mathbf{d}}_l$	7
$\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$	8
$\tilde{\mathfrak{a}}_l := \left(\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\sigma}_l, \tilde{\mathbf{z}}_l, \tilde{\mathbf{d}}_l\right)$	9
End	10
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$	11
$\mathbf{x} := \langle \tilde{\mathbf{x}} \rangle$	12
$\sigma := \langle \tilde{\sigma} \rangle$	13
$\mathbf{M} := \left(1 - \frac{1}{\tau_\mathrm{M}}\right)\mathbf{M} + \frac{1}{\tau_\mathrm{M}}\langle \tilde{\mathbf{d}}\tilde{\mathbf{z}}^\mathrm{T}\rangle$	14
Until(Termination_Condition)	15

- L4: $\tau = 1/\sqrt{2N}$ is asymptotically optimal for the sphere model
- L12f: recombine the *best* μ offspring' $\mathbf{x}$ and σ, according to Eq. (1/2)
- L14: update $\mathbf{M}$-matrix with learning rate

$$\tau_\mathrm{M} := 2 + \frac{(N+1)N}{\mu} \tag{18}$$

$$\langle \tilde{\mathbf{d}}\tilde{\mathbf{z}}^\mathrm{T}\rangle := \frac{1}{\mu}\sum_{m=1}^{\mu} \tilde{\mathbf{d}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^\mathrm{T} \tag{19}$$

- recommended truncation ratio: $\mu/\lambda = \frac{1}{4}$
- note, there are only two learning constants: τ and τ_M

Example: Optimization of a Lens Using σSA-MA-ES[6]

Objectives:

1. Find the optimal shape of glass body such that parallel incident light rays are concentrated in a given point P on a plane
2. Use a minimum of glass material possible (secondary goal)

General problem solving approach:

Step 1–3: system description, evaluation, and decision variables

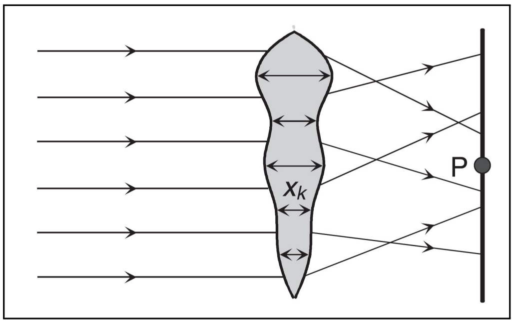


Figure 5: Evolvable glass body: incoming light rays from the left are refracted. Evolve thicknesses x_k such that the rays meet in P. The x_k ($k = 0, \ldots, K$) are the decision (aka control, aka objective) variables.

[6] Example adapted from *VDI Richtlinie 6224 Blatt 1*.

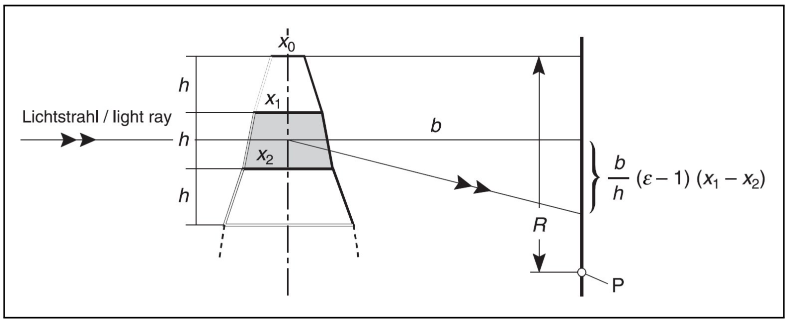


Figure 6: Lens is subdivided into trapezoidal slices of hight h; ϵ is refraction index.

- using the physical law of refraction on thin prisms (see Fig. 6), one can calculate the deviation Δ of the ray from focal point P on the plane considering the kth prism ($k = 1, \ldots, K$)

$$\Delta_k = R - \frac{h}{2} - (k-1)h - \frac{b}{h}(\epsilon - 1)(x_k - x_{k-1}), \qquad x_k \geq 0 \qquad (20)$$

Step 4: determine the goal (aka objective) function:

1. considering all K prisms, the squared sum can be used as measure for the optical image quality

$$f_{\text{focus}} := \sum_{k=1}^{K} \Delta_k^2 \qquad (21)$$

3. modeling the 2nd objective, the mass of the lense, we assume (for sake of simplicity) a mass density of one, therefore, the mass is simply the area of the lens (two-dimensional model)

$$f_{\text{mass}} := \sum_{k=1}^{K} h \frac{1}{2}(x_k + x_{k-1}) \tag{22}$$

4. note, we have *two objectives:*
   1. minimizing the optical image quality f_{focus}
   2. minimizing the mass of the lense f_{mass}

☞ actually, this calls for a *multi-objective evolutionary algorithm* (Pareto-optimization, however, beyond the scope of this tutorial)

5. instead, using *scalarization approach* where both objectives are combined in a weighted sum:

$$f_{\text{lens}}(x_0, \dots, x_K) := w f_{\text{focus}} + (1-w) f_{\text{mass}} \qquad w \in [0,1] \tag{23}$$

$$\text{and} \quad \forall k = 0, \dots, K: \; x_k \geq 0 \tag{24}$$

w controls the emphasis of optimization w.r.t focus ($w = 1$) or mass ($w = 0$)

Evolutionary Optimization of (23)

- $(\mu/\mu_I, \lambda)$-σSA-fMA-ES, Slide 16, is used with $N = K + 1$
- since $x_k \geq 0$ (thickness parameters!), the x_k in MA-ES must be transformed in order to be used in the objective function (23)
- that is: $f_{\text{lens}}(|x_0|, \dots, |x_K|)$ must be used for evaluation in Line 8 of MA-ES on Slide 16
- problem parameters: $\epsilon = 1.5$, $h = 1$, $K = 14$
- strategy parameters: $\mu = 5$, $\lambda = 20$[7], weighting factor $w = 0.9$[8]
- initialization: $x_k = 3$, $\sigma = 1$
- stopping criterion: mutation strength $\sigma < 10^{-5}$

[7] Recommended *truncation ratio* for MA-ES is $\mu/\lambda = 1/4$.
[8] Strong emphasis on optical quality.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matlab code of a simple Evolution Strategy applied to the optical lens
% optimization. Stategy type:  (mu/mu_I, lambda)-sigmaSA-MA-ES
% Scalarized minimization of quadratic focal point deviation and lens mass
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Copyright by Hans-Georg Beyer (HGB), 23.02.20. For non-commercial use
%% only. Commercial use requires written permission by Hans-Georg Beyer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global LensParms;               % physical parameters of the geometrical system
LensParms.h = 1; LensParms.b = 20; LensParms.R = 7; LensParms.eps = 1.5;
LensParms.d_init = 3;           % initial lens geometry (being rectangular)
global Weighting;               % scalarization factor for bi-objective problem
Weighting =.9;
n = 15;                    % number of free geometry parameters to be optimized
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here starts the MA-ES (cf. Pseudocode)
% initialization
mu = 5; lambda = 20;                                                  %  (L1)
x = LensParms.d_init*ones(n,1);                                       %  (L1)
sigma = 1; sigma_stop = 1e-5;                                         %  (L1)
M = eye(n);                                                           %  (L1)
tau = 1/sqrt(2*n); tau_M = 2 + n*(n+1)/mu;                            %  (L1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% here starts generation loop
while( sigma > sigma_stop )                                           %  (L2)
  for l=1:lambda                                                      %  (L3)
    sigmaTilde(l) = sigma * exp(tau*randn());                         %  (L4)
    zTilde(:, l) = randn(n,1);                                        %  (L5)
    dTilde(:, l) = M*zTilde(:, l);                                    %  (L6)
    xTilde(:, l) = x + sigmaTilde(l)*dTilde(:, l);                    %  (L7)
    fTilde(l) = f_lens(xTilde(:, l));                                 %  (L8)
  end
  [fsorted, r] = sort(fTilde, "ascend");                              % (L11)
  x = 1/mu * sum(xTilde(:, r(1:mu)), 2);                              % (L12)
  sigma = 1/mu * sum(sigmaTilde(r(1:mu)));                            % (L13)
  SUMds = zeros(n, n);                                                % (L14)
  for m=1:mu; SUMds = SUMds + dTilde(:, r(m))*zTilde(:, r(m))'; end;  % (L14)
  M = (1-1/tau_M)*M + (1/tau_M) * (1/mu)*SUMds;                       % (L14)
end                                                                   % (L15)
```

Figure 7: Matlab code of fast MA-ES for lens optimization.


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% after termination, abs(x) returns the x-coordinates (thicknesses)
% of the optimized lens geometry
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% here comes the objective function, to be saved in a .m-file
% we use abs(x) instead of x in order to ensure positiveness of parameters
function qual = f_lens(x)
  global LensParms Weighting;
  n = length(x);
  f_focus = sum( ( LensParms.R ...
                   - ( LensParms.h*((1:n-1)-.5) + LensParms.b/LensParms.h * ...
                       (LensParms.eps-1) * ...
                       (abs(x(2:n)) - abs(x(1:n-1)))' ) ).^2 );
  f_mass = LensParms.h*(sum(abs(x(2:n-1))) + 0.5*(abs(x(1))+abs(x(n))));
  qual = Weighting*f_focus + (1-Weighting)*f_mass; % weighting of goals
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 8: Matlab code of MA-ES for lens optimization continued: coding of the (aggregated) goal function f_{lens}, Eq. (23).

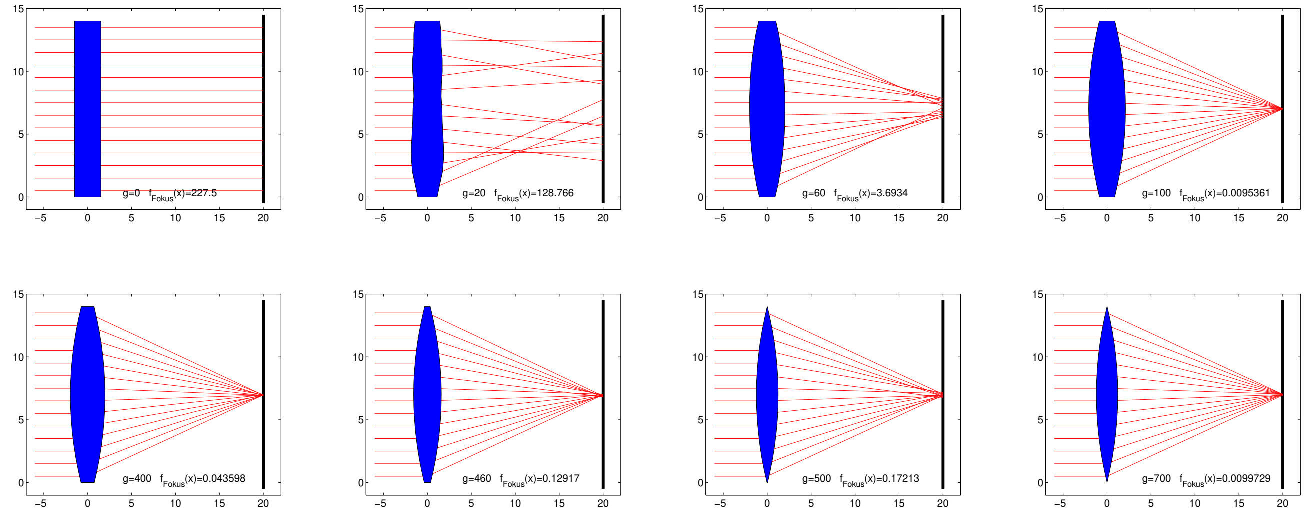


Figure 9: Snapshoots of the lens evolution using $(5/5_I, 20)$-σSA-MA-ES

- due to the choice of $w = 0.9$, at first the lens geometry evolves toward high optical quality
- after about 100 generations, the second goal (reducing the lens mass), dominates the evolution process resulting in defocussing
- at about generation 500, the lens has been reduced in mass and fine tuning of the image quality starts

On the dynamics of the evolution process

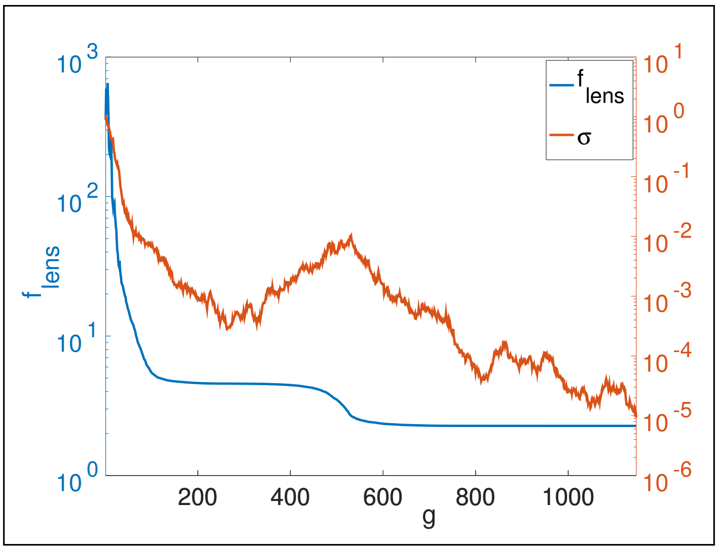


Figure 10: Evolutionary dynamics of the $(5/5_I, 20)$-σSA-MA-ES on the lens example.

- up to about generation $g = 100$, the evolution improves the focal quality
- then, the geometry must be rebuilt and therefore the $\mathbf{M}$-matrix, too
- after about $g = 300$ the $\mathbf{M}$ allows for larger mutation strengths σ
- finally, the mass of the lens reduces and the evolution converges

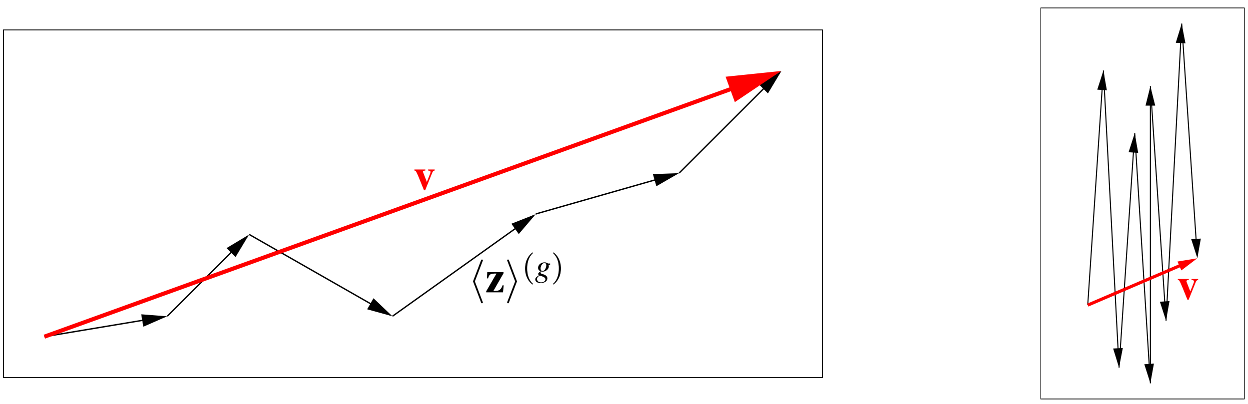
How to Get the Most Out

1. Path Cumulation
   1. learning promising evolution directions
   2. alternative σ control rule (cumulative step-size adaptation CSA)
2. Approximate Matrix-Vector Operations
   - limited memory MA-ES
3. Weighting the Individuals
   1. weighted recombination
   2. utilize the worst individuals (active matrix adaptation)

Path Cumulation

- consider the cumulation $\mathbf{v}$ of the generation history (g-generation counter) of the selected $\tilde{\mathbf{z}}$-vector centroids

$$\mathbf{v} := \sum_{g=g_1}^{g_1+G} \langle \mathbf{z} \rangle^{(g)} \qquad \text{where} \qquad \langle \mathbf{z} \rangle^{(g)} := \frac{1}{\mu} \sum_{m=1}^{\mu} \mathbf{z}_{m;\lambda}^{(g)} \tag{25}$$



$\mathbf{v}$-direction in $\mathbf{z}$-space with strong tendency versus weak tendency

Figure 11: Two qualitatively different paths $\mathbf{v}$ of concatenated $\langle \mathbf{z} \rangle^{(g)}$ centroids: Even though the average length of the $\langle \mathbf{z} \rangle$-vectors is larger for the right path than for the left, the cumulative effect is much larger for the left path indicating a preferred direction in $\mathbf{z}$-space.

☞ incorporate the $\mathbf{v}$ information in the update $\mathbf{M}$ update

- if the consecutive $\langle \mathbf{z} \rangle^{(g)}$ steps are uncorrelated it holds

$$\mathrm{E}[\mathbf{v}\mathbf{v}^{\mathrm{T}}] = \alpha \mathbf{I} \tag{26}$$

- however, the $\mathbf{v}\mathbf{v}^{\mathrm{T}}$-matrix could grow with the generation number g
- and past direction information may get stale after a while

☞ the cumulation of the $\langle \mathbf{z} \rangle$-vectors must be discounted by exponentially smoothing, leading to an $\mathbf{s}$-vector update

$$\mathbf{s} := (1 - \tfrac{1}{\tau_s})\mathbf{s} + \sqrt{\tfrac{\mu}{\tau_s}\left(2 - \tfrac{1}{\tau_s}\right)} \langle \mathbf{z} \rangle \tag{27}$$

☞ analogously to the Eq. (15) the $\mathbf{M}$-update one gets

$$\mathbf{M} := \mathbf{M}\left[\mathbf{I} + \tfrac{1}{\tau_1}(\mathbf{s}\mathbf{s}^{\mathrm{T}} - \mathbf{I})\right] \tag{28}$$

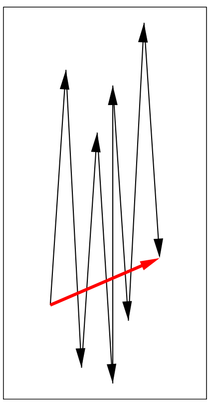
$$\tau_s = \Theta(N), \qquad \tau_1 = \Theta(N^2) \tag{29}$$

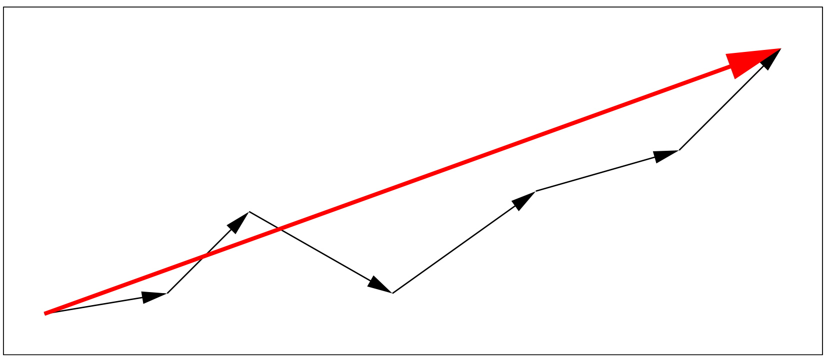
... and there is even more information in this v-path

- consider the length of the resulting path compared to single $\langle \mathbf{z} \rangle^{(g)}$ steps
- if there is no selection (flat fitness landscape) $\Rightarrow$ random path

➥ do not change mutation strength σ

- if there is selection and length of path is less than expected random path length, then decrease mutation strength σ





$\Rightarrow$ decrease $\qquad\qquad$ $\Rightarrow$ increase the step size (i.e., mutation strength)

- if there is selection and length of path is greater than expected random path length, then increase mutation strength σ

- it can be shown that this strategy is asymptotically optimal ($N \to \infty$) on the *static* sphere model (w/o noise)[9]
- since optimal mutation strength changes during the approach to the optimum, the steps used to calculate the statistics must be normalized w.r.t. the actual mutation strength σ
- this leads to the (modified[10]) cumulative step length adaptation (CSA) update rule[11]

$$\sigma := \sigma \exp\left[\frac{1}{2D}\left(\frac{\|\mathbf{s}\|^2}{N} - 1\right)\right] \qquad (30)$$

- single steps of the evolution path are very noisy, therefore, path length statistics must be updated by the weighted cumulation (27)
- damping constant $D = \Theta(\sqrt{N})$ (N – search space dimensionality)

[9] H.-G. Beyer and D.V. Arnold. *Qualms Regarding the Optimality of Cumulative Path Length Control in CSA/CMA-Evolution Strategies*. Evol. Comp., 11(1):19–28, 2003.

[10] D.V. Arnold, H.-G. Beyer. *Performance Analysis of Evolutionary Optimization With Cumulative Step Length Adaptation*. IEEE Trans. on Autom. Control, 49(4): 617–622, 2004.

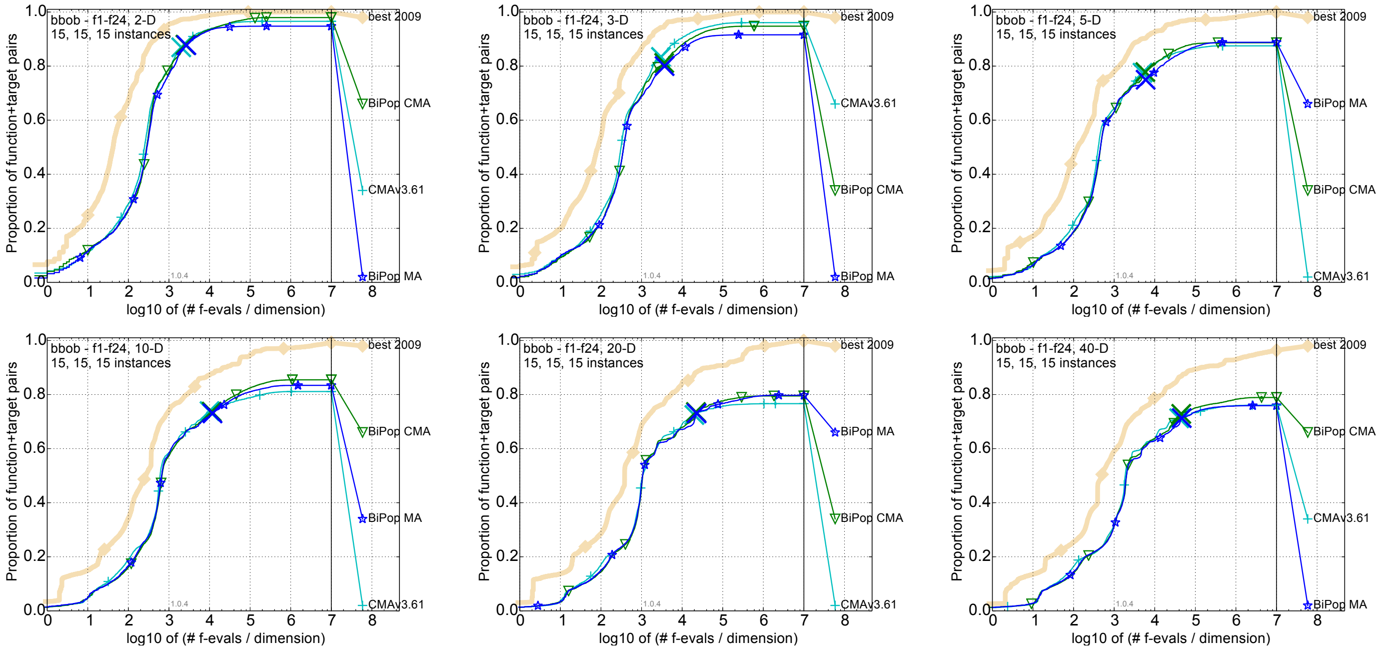
[11] N. Hansen, A. Ostermeier. *Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation*. In Proc. 1996 IEEE Int'l Conf. on Evol. Comp. (ICEC'96), 312–317.

Putting Things Together: The $(\mu/\mu_I, \lambda)$-MA-ES

Initialize $(\mathbf{x}, \sigma, D, \tau_s, \tau_1, \tau_M, \mathbf{s} := \mathbf{1}, \mathbf{M} := \mathbf{I})$	(M1)
Repeat	(M2)
For $l := 1$ **To** λ	(M3)
$\tilde{\mathbf{z}}_l := \mathcal{N}_l(\mathbf{0}, \mathbf{I})$	(M4)
$\tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l$	(M5)
$\tilde{\mathbf{x}}_l := \mathbf{x} + \sigma\tilde{\mathbf{d}}_l$	(M6)
$\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$	(M7)
$\tilde{\mathfrak{a}}_l := (\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l)$	(M8)
End	(M9)
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$	(M10)
$\mathbf{x} := \langle\tilde{\mathbf{x}}\rangle$	(M11)
$\mathbf{s} := \left(1 - \frac{1}{\tau_s}\right)\mathbf{s} + \sqrt{\frac{\mu}{\tau_s}\left(2 - \frac{1}{\tau_s}\right)}\langle\tilde{\mathbf{z}}\rangle$	(M12)
$\mathbf{M} := \mathbf{M}\left[\mathbf{I} + \frac{1}{\tau_1}(\langle\mathbf{s}\mathbf{s}^\mathrm{T}\rangle - \mathbf{I}) + \frac{1}{\tau_\mathrm{M}}(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^\mathrm{T}\rangle - \mathbf{I})\right]$	(M13)
$\sigma := \sigma \exp\left[\frac{1}{2D}\left(\frac{\|\mathbf{s}\|^2}{N} - 1\right)\right]$	(M14)
Until(Termination_Condition)	(M15)
Return$(\mathbf{x})$	(M16)

- M6: there is only one σ
- M13: incorporation of the $\mathbf{s}$-path direction
- M1: initial $\mathbf{s} := (1, \ldots, 1)^\mathrm{T} \in \mathbb{R}^N$
- M12: $\mathbf{s}$-path cumulation
- M14: σ-update using cumulative step length adapatation (CSA)
- strategy parameters (can be improved):
  - $D := \sqrt{N}$
  - $\tau_1 := 2N^2$
  - $\tau_s := N$
  - $\tau_M := 2 + \frac{(N+1)N}{\mu}$
- published in TEVC 21(5), see Footnote 4

- the MA-ES can be regarded as an *approximation* of the CMA-ES
- however, the performance differences on standard test beds including the COCO BBOB are not really significant (using weighted recombination)
- no big differences even for population sizes such as $\lambda = 4N^2$
- **Example:** COCO BBOB performance in BiPop-ES setting:



Reducing the Internal Algorithm's Costs: The Fast-$(\mu/\mu_I, \lambda)$-MA-ES

Initialize $(\mathbf{x}, \sigma, D, \tau_s, \tau_1, \tau_M, \mathbf{s} := \mathbf{1}, \mathbf{M} := \mathbf{I})$ (M1)
Repeat (M2)
  For $l := 1$ **To** λ (M3)
    $\tilde{\mathbf{z}}_l := \mathcal{N}_l(\mathbf{0}, \mathbf{I})$ (M4)
    $\tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l$ (M5)
    $\tilde{\mathbf{x}}_l := \mathbf{x} + \sigma\tilde{\mathbf{d}}_l$ (M6)
    $\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$ (M7)
    $\tilde{\mathfrak{a}}_l := (\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l, \tilde{\mathbf{d}}_l)$ (M8)
  End (M9)
  RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$ (M10)
  $\mathbf{x} := \langle\tilde{\mathbf{x}}\rangle$ (M11)
  $\mathbf{s} := \left(1 - \frac{1}{\tau_s}\right)\mathbf{s} + \sqrt{\frac{\mu}{\tau_s}\left(2 - \frac{1}{\tau_s}\right)}\langle\tilde{\mathbf{z}}\rangle$ (M12)
  $\mathbf{M} := \left(1 - \frac{1}{\tau_1} - \frac{1}{\tau_M}\right)\mathbf{M} + \frac{1}{\tau_1}\langle(\mathbf{M}\mathbf{s})\mathbf{s}^{\mathrm{T}}\rangle + \frac{1}{\tau_{\mathrm{M}}}\langle\tilde{\mathbf{d}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle$ (M13)
  $\sigma := \sigma \exp\left[\frac{1}{2D}\left(\frac{\|\mathbf{s}\|^2}{N} - 1\right)\right]$ (M14)
Until(Termination_Condition) (M15)
Return$(\mathbf{x})$ (M16)

- M13 dominates internal cost of the algorithm, Slide 30: $\mathcal{O}(N^3)$ (matrix-matrix multiplication)
- reordering M13 yields $\mathcal{O}(N^2)$ since there are only matrix-vector products and sums
- note, this algorithm is *equivalent* to the $(\mu/\mu_I, \lambda)$-MA-ES of Slide 30
- now, the λ $\tilde{\mathbf{d}}$ calculations in (M5) become the bottleneck

Advantages of the MA-ES

Using MA-ES instead of CMA-ES is recommended, because:

1. simpler implementation, no eigenvalue or Cholesky decomposition, no Cholesky factorization (faster than the KRAUSE ET AL.[12] approach)
2. . . . and better suited for GPUs
3. uses only one evolution path, thus, reduced number of strategy parameters
4. greater numerical stability, regularization yet possible
5. due to its simplicity, the MA-ES is a starting point for the derivation of *approximation schemes* for (M5, M13) to further reduce the single offspring generation cost (see Slide 35ff)
6. (M13) should be the starting point for theoretical convergence analyses
7. MA-ES might be easier for teaching undergraduates

[12] O. Krause, D.R. Arbones, and C. Igel. *CMA-ES with Optimal Covariance Update and Storage Complexity*, in Proc. Adv. Neural Inf. Process. Syst. 29 (NIPS'2016), pp. 370–378, Barcelona, Spain, 2016.

Example: Cholesky-CMA-ES

Algorithm 1: The Cholesky-CMA-ES.

input : λ, μ, $\boldsymbol{m}_1$, $\omega_{i=1\ldots\mu}$, c_σ, d_σ, c_c, c_1 and c_μ

$A_1 = \mathrm{I}$, $p_{c,1} = \mathbf{0}$, $p_{\sigma,1} = \mathbf{0}$

for $t = 1, 2, \ldots$ **do**

  for $i = 1, \ldots, \lambda$ **do**

    $\boldsymbol{x}_{i,t} = \sigma_t A_t \boldsymbol{y}_{i,t} + \boldsymbol{m}_t,\ \boldsymbol{y}_{i,t} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$

  Sort $\boldsymbol{x}_{i,t}$, $i = 1, \ldots, \lambda$ increasing by $f(\boldsymbol{x}_{i,t})$

  $\boldsymbol{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \boldsymbol{x}_{i,t}$

  $\boldsymbol{p}_{c,t+1} = (1 - c_c)\boldsymbol{p}_{c,t} + \sqrt{c_c(2 - c_c)\mu_{\mathrm{eff}}} \frac{\boldsymbol{m}_{t+1} - \boldsymbol{m}_t}{\sigma_t}$

  `// Apply formula (2) to` A_t

  $A_{t+1} \leftarrow \sqrt{1 - c_1 - c_\mu} A_t$

  $A_{t+1} \leftarrow \mathrm{rankOneUpdate}(A_{t+1}, c_1, \boldsymbol{p}_{c,t+1})$

  for $i = 1, \ldots, \mu$ **do**

    $A_{t+1} \leftarrow \mathrm{rankOneUpdate}(A_{t+1}, c_\mu \omega_i, \frac{\boldsymbol{x}_{i,t} - \boldsymbol{m}_t}{\sigma_t})$

  `// Update` σ `using` $\hat{\boldsymbol{s}}_k$ `as in (5)`

  $\boldsymbol{p}_{\sigma,t+1} = (1 - c_\sigma)\boldsymbol{p}_{\sigma,t} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}} A_t^{-1} \frac{\boldsymbol{m}_{t+1} - \boldsymbol{m}_t}{\sigma_t}$

  $\sigma_{t+1} = \sigma_t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_{\sigma,t+1}\|}{\mathbb{E}\{\chi\}} - 1\right)\right)$

Algorithm 2: rankOneUpdate$(A, \beta, \boldsymbol{v})$

input : Cholesky factor $A \in \mathbb{R}^{d \times d}$ of C, $\beta \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^d$

output : Cholesky factor A' of $C + \beta \boldsymbol{v}\boldsymbol{v}^T$

$\boldsymbol{\alpha} \leftarrow \boldsymbol{v}$

$b \leftarrow 1$

for $j = 1, \ldots, d$ **do**

  $A'_{jj} \leftarrow \sqrt{A_{jj}^2 + \frac{\beta}{b}\alpha_j^2}$

  $\gamma \leftarrow A_{jj}^2 b + \beta\alpha_j^2$

  for $k = j + 1, \ldots, d$ **do**

    $\alpha_k \leftarrow \alpha_k - \frac{\alpha_j}{A_{jj}} A_{kj}$

    $A'_{kj} = \frac{A'_{jj}}{A_{jj}} A_{kj} + \frac{A'_{jj}\beta\alpha_j}{\gamma}\alpha_k$

  $b \leftarrow b + \beta\frac{\alpha_j^2}{A_{jj}^2}$

Published in: O. Krause, D.R. Arbones, and C. Igel, "CMA-ES with optimal covariance Update and Storage Complexity," in *Proc. Adv. Neural Inf. Process. Syst.* Barcelona, Spain, 2016, pp. 370–378.

... for large search space dimensionalities N:

The Limited Memory MA-ES - the LM-MA-ES[13]

- algorithm complexity of the fast MA-ES is still of $\mathcal{O}(N^2)$ due to (M11) and (M5)
- if one wants to reduce the complexity further, one needs to approximate the matrix-vector operations in (M5) and (M11)
- an approach taken ideas from *Limited Memory BFGS* comes into mind, however, an alternative approach will be considered here
- in order to approximate the **M** matrix, γ vectors $\mathbf{p}_k$ are used
- running γ evolution paths at different time scales

[13] I. Loshchilov, T. Glasmachers, and H.-G. Beyer. Large Scale Black-box Optimization by Limited-Memory Matrix Adaptation. *IEEE Transactions on Evolutionary Computation*, 2018. DOI: 10.1109/TEVC.2018.2855049.

$(\mu/\mu_w, \lambda)$-LM-MA-ES

$$
\begin{array}{ll}
\textbf{Initialize}\left(\mathbf{x}^{(0)}, \sigma^{(0)}, g := 0, \mathbf{s}^{(0)} := \mathbf{1}, \mathbf{p}^{(0)}_{1\ldots\gamma} := \mathbf{0}\right) & \text{(L1)}\\
\textbf{Repeat} & \text{(L2)}\\
\quad \textbf{For}\ \ l := 1\ \ \textbf{To}\ \ \lambda & \text{(L3)}\\
\quad\quad \tilde{\mathbf{z}}_l^{(g)} := \boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I}) & \text{(L4)}\\
\quad\quad \tilde{\mathbf{d}}_l^{(g)} := \tilde{\mathbf{z}}_l^{(g)} & \text{(L5)}\\
\quad\quad \textbf{For}\ \ k := 1\ \ \textbf{To}\ \ \min(g, \gamma) & \text{(L6)}\\
\quad\quad\quad \tilde{\mathbf{d}}_l^{(g)} := (1 - c_{d,k})\tilde{\mathbf{d}}_l^{(g)} + c_{d,k}\left(\mathbf{p}_k^{(g)\mathrm{T}}\tilde{\mathbf{d}}_l^{(g)}\right)\mathbf{p}_k^{(g)} & \text{(L7)}\\
\quad\quad \textbf{End} & \text{(L8)}\\
\quad\quad \tilde{\mathbf{x}}_l^{(g)} := \mathbf{x}^{(g)} + \sigma^{(g)}\tilde{\mathbf{d}}_l^{(g)} & \text{(L9)}\\
\quad\quad \tilde{f}_l^{(g)} := f\left(\tilde{\mathbf{x}}_l^{(g)}\right) & \text{(L10)}\\
\quad\quad \tilde{\mathfrak{a}}_l^{(g)} := \left(\tilde{f}_l^{(g)}, \tilde{\mathbf{x}}_l^{(g)}, \tilde{\mathbf{z}}_l^{(g)}\right) & \text{(L11)}\\
\quad \textbf{End} & \text{(L12)}\\
\quad \textbf{RankOffspringPopulation}\left(\tilde{\mathfrak{a}}_1^{(g)}, \ldots, \tilde{\mathfrak{a}}_\lambda^{(g)}\right) & \text{(L13)}\\
\quad \mathbf{x}^{(g+1)} := \left\langle \tilde{\mathbf{x}}^{(g)} \right\rangle_w & \text{(L14)}\\
\quad \mathbf{s}^{(g+1)} := (1 - c_s)\mathbf{s}^{(g)} + \sqrt{\mu_{\mathrm{eff}} c_s (2 - c_s)}\left\langle \tilde{\mathbf{z}}^{(g)} \right\rangle_w & \text{(L15)}\\
\quad \textbf{For}\ \ k := 1\ \ \textbf{To}\ \ \gamma & \text{(L16)}\\
\quad\quad \mathbf{p}_k^{(g+1)} := (1 - c_{p,k})\mathbf{p}_k^{(g)} + \sqrt{\mu_{\mathrm{eff}} c_{p,k} (2 - c_{p,k})}\left\langle \tilde{\mathbf{z}}^{(g)} \right\rangle_w & \text{(L17)}\\
\quad \textbf{End} & \text{(L18)}\\
\quad \sigma^{(g+1)} := \sigma^{(g)} \exp\left[\frac{c_s}{2}\left(\frac{\|\mathbf{s}^{(g+1)}\|^2}{N} - 1\right)\right] & \text{(L19)}\\
\quad g := g + 1 & \text{(L20)}\\
\textbf{Until}(\text{termination condition(s) fulfilled}) & \text{(L21)}
\end{array}
$$

heuristically chosen strategy parameters (for $N > 50$):

- number of evolution paths $\gamma := 4 + \lfloor 3 \ln N \rfloor$
- weighting constants $c_{d,k} := \frac{1}{1.5^{k-1} N}$
- σ-evolution path cumulation constant $c_s := \frac{2\lambda}{N} \quad (< \frac{1}{2})$
- **p**-evolution path cumulation constants $c_{p,k} := \frac{\lambda}{4^{k-1} N}$

using *weighted recombination*: $\langle\cdot\rangle_w$ and μ_{eff}, see Slide 38

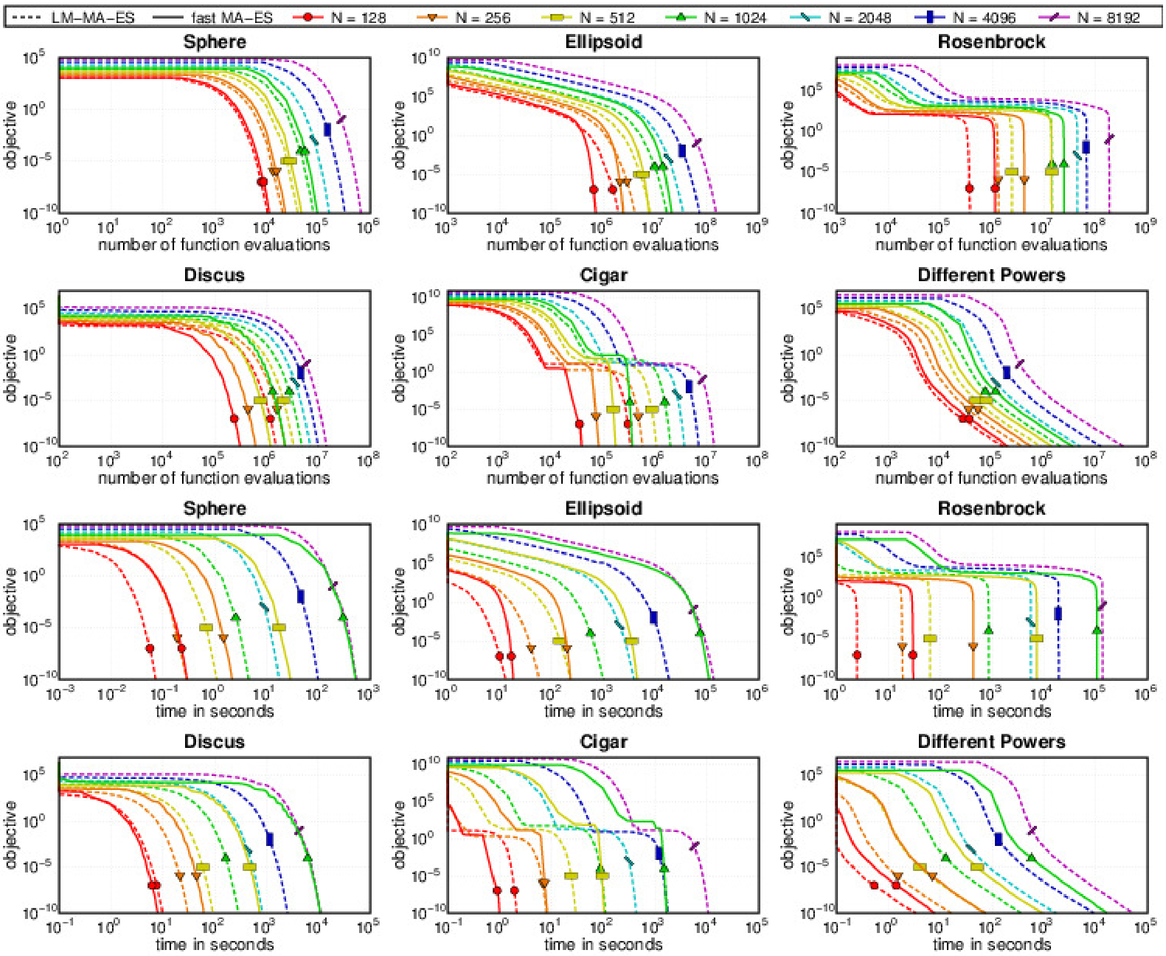


Figure 12: Performance of LM-MA-ES vs. fastMA-ES w.r.t. #-fevals and time.

Remarks (LM-MA-ES Performance)

- algorithm complexity of LM-MA-ES: $\mathcal{O}(\gamma N) = \mathcal{O}(N \ln N)$
- note, there are problem instances where LM-MA-ES outperforms MA-ES w.r.t. function evaluations (e.g. Rosenbrock $N = 256, 512$)

Remarks (additional details LM-MA-ES)

- *weighted* recombination: $\left\langle \tilde{\mathbf{a}}^{(g)} \right\rangle_w := \sum_{m=1}^{\mu} w_m \mathbf{a}_{m;\lambda}^{(g)}$
- "effective" parent population value $\mu_{\text{eff}} = \left(\sum_{m=1}^{\mu} w_m^2\right)^{-1}$
- weights w_m must fulfill $\sum_{m=1}^{\mu} w_m \stackrel{!}{=} 1$ and should not emphasize bad individuals, e.g.

$$w_m := \begin{cases} \dfrac{\ln\left(\frac{\lambda+1}{2}\right) - \ln m}{\sum_{k=1}^{\mu}\left(\ln\left(\frac{\lambda+1}{2}\right) - \ln k\right)}, & \text{for } 1 \le m \le \mu, \\ 0, & \text{otherwise} \end{cases}$$

- population sizing: $\lambda = 4 + \lfloor 3 \ln N \rfloor$ and $\mu = \left\lfloor \frac{\lambda}{2} \right\rfloor$

Note, these recommendations are directly taken from: N. Hansen. *The CMA Evolution Strategy: A Comparing Review.* DOI: 10.1007/3-540-32494-1_4

Design Principles for MA-ES on Constrained Problems

Optimization under constraints is a wide and almost uncharted field for Matrix Adaptation Evolution Strategies

1. equality contraints

$$\forall j = 1, \ldots, J: \; h_j(\mathbf{x}) = 0, \quad \mathbf{x} \in \mathbb{R}^{N_x} \tag{31}$$

2. inequality constraints

$$\forall k = 1, \ldots, K: \; g_k(\mathbf{x}) \leq 0, \quad \mathbf{x} \in \mathbb{R}^{N_x} \tag{32}$$

3. mixtures of (31) and (32)

Often used: Penalty methods that do not touch the ES itself, but shifts the problem into a modified objective function

Up until recently, there were only a few exeptions from this approach, most notable the work of D. Arnold et al.[14]

[14] E.g.: D.V. Arnold, *Reconsidering constraint release for active-set evolution strategies,* GECCO'17, pp. 665–672. DOI: 10.1145/3071178.3071294

Equality Constraints

often used approach:

- turn (31) into inequalities and use methods for inequality handling (standard way in EAs)

$$h_j(\mathbf{x}) = 0 \;\Leftrightarrow\; |h_j(\mathbf{x})| - \delta \leq 0 \quad \text{for} \quad \delta \to 0 \tag{33}$$

- **problem:** δ must be chosen sufficiently small

➥ feasible region is very small (measure goes to zero if $\delta \to 0$)

- **ideal solution:** generate offspring that fulfill $h_j(\mathbf{x}) = 0$ automatically

  Options:

  1. find suitable transformation that transforms a "raw offspring" into a feasible offspring

     ☞ applicable for special $h_j(\mathbf{x})$ cases only

  2. repair "raw offspring" such that it fulfills (31) and (optionally) perform a back-calculation to adapt the **M** matrix

☞ these methods are referred to as *inner point methods*

1. Transformation methods

1. linear equality constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \text{i.e.,} \quad h_j(\mathbf{x}) = \sum_{n=1}^{N} (\mathbf{A})_{jn}(\mathbf{x})_n - (\mathbf{b})_j = 0 \qquad (34)$$

- use null-space mutations[15]
- note that any solution $\mathbf{x}$ of (34) can be decomposed into an *in*homogenous and a homogenous solution

$$\mathbf{A}\left(\mathbf{x}_{\text{inh}} + \mathbf{x}_{\text{h}}\right) = \mathbf{b} \quad \text{where} \quad \mathbf{A}\mathbf{x}_{\text{h}} = \mathbf{0}, \quad \text{i.e.,} \quad \mathbf{x}_{\text{h}} \in \text{null}(\mathbf{A}) \qquad (35)$$

- the elements of the null space of $\mathbf{A}$ can be represented by an orthogonal basis, the vectors of this basis can be collected in a matrix $\mathbf{B} \in \mathbb{R}^{N_x \times N}$ obeying

$$\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{I} \quad \text{and} \quad \mathbf{A}\mathbf{B} = \mathbf{0}, \qquad (36)$$

- the initial parental state is obained by solving the linear system $\mathbf{A}\mathbf{x}_{\text{inh}} = \mathbf{b}$ (perhaps adding an $\mathbf{x}_{\text{h}}$ to shift the initial parent to a desired position)

[15] First introduced in: P. Spettel et al.: “A Covariance Matrix Self-Adaptation Evolution Strategy for Optimization under Linear Constraints.” *IEEE Transactions on Evolutionary Computation* 23(3):514–524, 2019. DOI: 10.1109/TEVC.2018.2871944

Initialize$\left(\mathbf{x} := \mathbf{x}_{\text{inh}}, \sigma, D, \tau_s, \tau_1, \tau_M,\right.$	
$\left.\mathbf{s} := \mathbf{1}, \mathbf{M} := \mathbf{I}, \mathbf{B} := \text{null}(\mathbf{A})\right)$	(M1)
Repeat	(M2)
For $l := 1$ **To** λ	(M3)
$\tilde{\mathbf{z}}_l := \mathcal{N}_l(\mathbf{0}, \mathbf{I})$	(M4)
$\tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l$	(M5)
$\tilde{\mathbf{x}}_l := \mathbf{x} + \sigma\mathbf{B}\tilde{\mathbf{d}}_l$	(M6)
$\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$	(M7)
$\tilde{\mathfrak{a}}_l := \left(\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l\right)$	(M8)
End	(M9)
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$	(M10)
$\mathbf{x} := \langle\tilde{\mathbf{x}}\rangle$	(M11)
$\mathbf{s} := \left(1 - \frac{1}{\tau_s}\right)\mathbf{s} + \sqrt{\frac{\mu}{\tau_s}\left(2 - \frac{1}{\tau_s}\right)}\langle\tilde{\mathbf{z}}\rangle$	(M12)
$\mathbf{M} := \mathbf{M}\left[\mathbf{I} + \frac{1}{\tau_1}\left(\langle\mathbf{s}\mathbf{s}^{\mathrm{T}}\rangle - \mathbf{I}\right) + \frac{1}{\tau_{\mathrm{M}}}\left(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle - \mathbf{I}\right)\right]$	(M13)
$\sigma := \sigma \exp\left[\frac{1}{2D}\left(\frac{\Vert\mathbf{s}\Vert^2}{N} - 1\right)\right]$	(M14)
Until(Termination_Condition)	(M15)
Return$(\mathbf{x})$	(M16)

- M1: initial $\mathbf{x}$ is obtained by solving $\mathbf{A}\mathbf{x}_{\text{inh}} = \mathbf{b}$
- dimensionality N of $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{d}}$ is given by $N := \text{rank}(\mathbf{B})$
- $\mathbf{M} \in \mathbb{R}^{N \times N}$

2. ellipsoidal equality constraint

$$\mathbf{x}^{\mathrm{T}}\mathbf{S}\mathbf{x} = \kappa > 0, \quad \forall \mathbf{x} \in \mathbb{R}^N \wedge \mathbf{x} \neq \mathbf{0} \tag{37}$$

- use non-linear transformation[16]
- consider the Cholesky decomposition of $\mathbf{S}$ in the $\mathbf{A}$-factor

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{S} \tag{38}$$

- then an offspring $\tilde{\mathbf{x}}$ satisfying (37) is obtained by

$$\tilde{\mathbf{x}} := \sqrt{\kappa}\frac{\mathbf{x} + \sigma\mathbf{A}^{-1}\tilde{\mathbf{d}}}{\|\mathbf{A}\mathbf{x} + \sigma\tilde{\mathbf{d}}\|} \tag{39}$$

- the parental state can be obtained similarly

$$\langle\tilde{\mathbf{x}}\rangle := \sqrt{\kappa}\frac{\mathbf{x} + \sigma\mathbf{A}^{-1}\langle\tilde{\mathbf{d}}\rangle}{\|\mathbf{A}\mathbf{x} + \sigma\langle\tilde{\mathbf{d}}\rangle\|} \tag{40}$$

[16] First introduced in: P. Spettel & H.-G. Beyer: “Matrix Adaptation Evolution Strategies for Optimization Under Nonlinear Equality Constraints.” *Swarm and Evolutionary Computation*, 2019. DOI: 10.1016/j.swevo.2020.100653

$$
\begin{array}{ll}
\text{Initialize}\big(\mathbf{x}, \sigma, D, \tau_s, \tau_1, \tau_M, \mathbf{s} := \mathbf{1}, \mathbf{M} := \mathbf{I}, & \\
\quad \mathbf{A} := \text{CholeskyDecomposition}(\mathbf{S})\big) & \text{(M1)} \\
\textbf{Repeat} & \text{(M2)} \\
\quad \textbf{For } l := 1 \textbf{ To } \lambda & \text{(M3)} \\
\quad\quad \tilde{\mathbf{z}}_l := \mathcal{N}_l(\mathbf{0}, \mathbf{I}) & \text{(M4)} \\
\quad\quad \tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l & \text{(M5)} \\
\quad\quad \tilde{\mathbf{x}}_l := \sqrt{\kappa}\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\tilde{\mathbf{d}}_l}{\|\mathbf{A}\mathbf{x}+\sigma\tilde{\mathbf{d}}_l\|} & \text{(M6)} \\
\quad\quad \tilde{f}_l := f(\tilde{\mathbf{x}}_l) & \text{(M7)} \\
\quad\quad \tilde{\mathfrak{a}}_l := \left(\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l\right) & \text{(M8)} \\
\quad \textbf{End} & \text{(M9)} \\
\quad \text{RankOffspringPopulation}(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda) & \text{(M10)} \\
\quad \mathbf{x} := \sqrt{\kappa}\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\langle\tilde{\mathbf{d}}\rangle}{\|\mathbf{A}\mathbf{x}+\sigma\langle\tilde{\mathbf{d}}\rangle\|} & \text{(M11)} \\
\quad \mathbf{s} := \left(1 - \frac{1}{\tau_s}\right)\mathbf{s} + \sqrt{\frac{\mu}{\tau_s}\left(2 - \frac{1}{\tau_s}\right)}\langle\tilde{\mathbf{z}}\rangle & \text{(M12)} \\
\quad \mathbf{M} := \mathbf{M}\left[\mathbf{I} + \frac{1}{\tau_1}\left(\langle\mathbf{s}\mathbf{s}^{\mathrm{T}}\rangle - \mathbf{I}\right) + \frac{1}{\tau_{\mathrm{M}}}\left(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle - \mathbf{I}\right)\right] & \text{(M13)} \\
\quad \sigma := \sigma\exp\left[\frac{1}{2D}\left(\frac{\|\mathbf{s}\|^2}{N} - 1\right)\right] & \text{(M14)} \\
\textbf{Until}(\text{Termination\_Condition}) & \text{(M15)} \\
\textbf{Return}(\mathbf{x}) & \text{(M16)}
\end{array}
$$

- M6: non-linear transformation of direction vector
- M1: $\mathbf{A}^{-1}$ can be calculated as well
- M11: this transformation might be replaced by $\langle\mathbf{x}\rangle$, in that case M16 must return $\tilde{\mathbf{x}}_{1;\lambda}$. Note, in that case performance might/will be different

Remark: There is also a transformation (not discussed here) that satisfies hyperbolic constraints

2. Repair method

- even if the parental state $\mathbf{x}$ fulfills $\forall j = 1, \ldots, J: \ h_j(\mathbf{x}) = 0$, the offspring state $\tilde{\mathbf{x}} := \mathbf{x} + \sigma\tilde{\mathbf{d}}$ will (almost surely) violate the constraint(s)

$$\mathbf{h}(\tilde{\mathbf{x}}) \neq \mathbf{0} \tag{41}$$

☞ $\tilde{\mathbf{x}}$ must be repaired by adding $\Delta\mathbf{x}$ such that

$$\mathbf{h}(\tilde{\mathbf{x}} + \Delta\mathbf{x}) = \mathbf{0} \tag{42}$$

- Taylor expansion yields with the Jacobian matrix $(\mathbf{J})_{jn} := \frac{\partial h_j}{\partial x_n}$

$$\mathbf{h}(\tilde{\mathbf{x}} + \Delta\mathbf{x}) = \mathbf{h}(\tilde{\mathbf{x}}) + \mathbf{J}\Delta\mathbf{x} + \ldots = \mathbf{0} \tag{43}$$

- neglecting higher-order terms, $\Delta\mathbf{x}$ can be approximately determined using the MOORE-PENROSE *Pseudoinverse* $\mathbf{J}^\dagger$, one obtains

$$\Delta\mathbf{x} = -\mathbf{J}^\dagger(\tilde{\mathbf{x}})\mathbf{h}(\tilde{\mathbf{x}}) \tag{44}$$

- thus, performing the update $\tilde{\mathbf{x}} := \tilde{\mathbf{x}} + \Delta\mathbf{x}$ yielding the iterative scheme

$$\tilde{\mathbf{x}} := \tilde{\mathbf{x}} - \mathbf{J}^\dagger(\tilde{\mathbf{x}})\mathbf{h}(\tilde{\mathbf{x}}) \tag{45}$$

- the scheme (44) is iterated until $\|\mathbf{h}(\tilde{\mathbf{x}})\|$ is sufficiently small (e.g. 10^{-8})
- this process is performed by the function $\text{Repair}(\tilde{\mathbf{x}}, \mathbf{h})$ in the pseudocode on the next slide[17]
- the Jacobian can be determined numerically (black-box scenario) or symbolically (white-box)

MA Peculiarities:

- $\tilde{\mathbf{z}}$ back calculation to increase probability for individuals in vicinity of constraint hypersurface
- requires the "inverse" $\mathbf{M}_{\text{inv}}$ of $\mathbf{M}$
- can be done by an $\mathbf{M}_{\text{inv}}$ update (initially $\mathbf{M}_{\text{inv}} = \mathbf{I}$)

$$\mathbf{M}_{\text{inv}} := \left[\mathbf{I} - \frac{1}{\tau_1}\left(\langle\mathbf{s}\mathbf{s}^{\text{T}}\rangle - \mathbf{I}\right) - \frac{1}{\tau_{\text{M}}}\left(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\text{T}}\rangle - \mathbf{I}\right)\right] \mathbf{M}_{\text{inv}} \tag{46}$$

[17] For details it is referred to: P. Spettel & H.-G. Beyer: "Matrix Adaptation Evolution Strategies for Optimization Under Nonlinear Equality Constraints." *Swarm and Evolutionary Computation*, 2019. DOI: 10.1016/j.swevo.2020.100653

Initialize $(\mathbf{x}, \sigma, D, \tau_s, \tau_1, \tau_M, \mathbf{s} := \mathbf{1}, \mathbf{M} := \mathbf{I})$	(M1)
Repeat	(M2)
For $l := 1$ **To** λ	(M3)
$\tilde{\mathbf{z}}_l := \mathcal{N}_l(\mathbf{0}, \mathbf{I})$	(M4)
$\tilde{\mathbf{d}}_l := \mathbf{M}\tilde{\mathbf{z}}_l$	(M5)
$\tilde{\mathbf{x}}_l :=$ Repair$(\mathbf{x} + \sigma\tilde{\mathbf{d}}_l, \mathbf{h})$	(M6)
$\tilde{\mathbf{z}}_l := \frac{1}{\sigma}\mathbf{M}_{\mathrm{inv}}(\tilde{\mathbf{x}}_l - \mathbf{x})$	(M7)
$\tilde{f}_l := f(\tilde{\mathbf{x}}_l)$	(M8)
$\tilde{\mathfrak{a}}_l := (\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l)$	(M9)
End	(M10)
RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$	(M11)
$\mathbf{x} :=$ Repair$(\langle\tilde{\mathbf{x}}, \mathbf{h}\rangle)$	(M12)
$\mathbf{s} := \left(1 - \frac{1}{\tau_s}\right)\mathbf{s} + \sqrt{\frac{\mu}{\tau_s}\left(2 - \frac{1}{\tau_s}\right)}\langle\tilde{\mathbf{z}}\rangle$	(M13)
$\mathbf{M} := \mathbf{M}\left[\mathbf{I} + \frac{1}{\tau_1}(\langle\mathbf{s}\mathbf{s}^{\mathrm{T}}\rangle - \mathbf{I}) + \frac{1}{\tau_{\mathrm{M}}}(\langle\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{\mathrm{T}}\rangle - \mathbf{I})\right]$	(M14)
$\sigma := \sigma \exp\left[\frac{1}{2D}\left(\frac{\lVert\mathbf{s}\rVert^2}{N} - 1\right)\right]$	(M15)
Until(Termination_Condition)	(M16)
Return($\mathbf{x}$)	(M17)

- M6: by iterating (45)
- M7: calculate back such that $\tilde{\mathbf{z}}_l$ fulfills $\mathbf{h}(\tilde{\mathbf{x}}_l) = \mathbf{0}$
- $\mathbf{M}_{\mathrm{inv}}$ – "inverse matrix" either by pseudoinverse of $\mathbf{M}$ or iteration after (M14) using Eq. (46)
- M12: this transformation might be replaced by $\langle\mathbf{x}\rangle$, in that case M16 must return $\tilde{\mathbf{x}}_{1;\lambda}$. Note, in that case performance might/will be different
- Note, this MA-ES in an *interior point method*

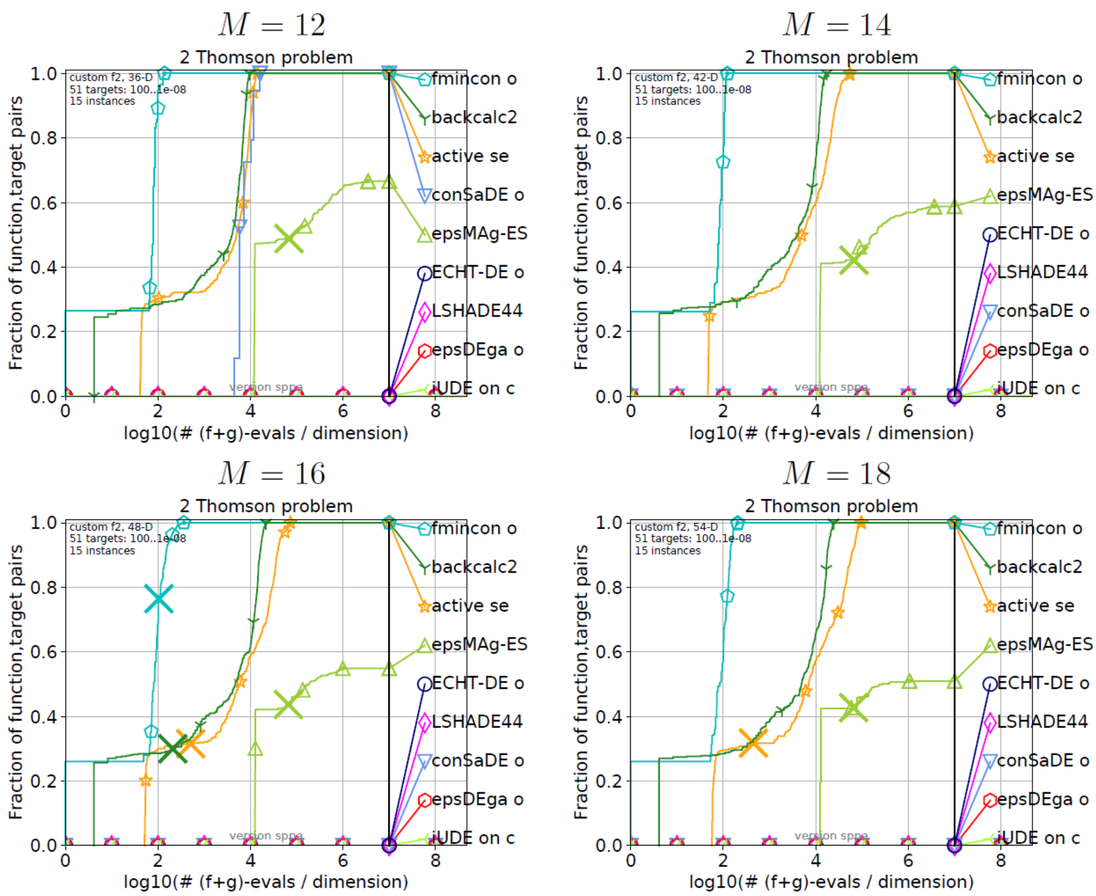


Figure 13: ECDF-plots of Thomson's problem (M – number of points on sphere).

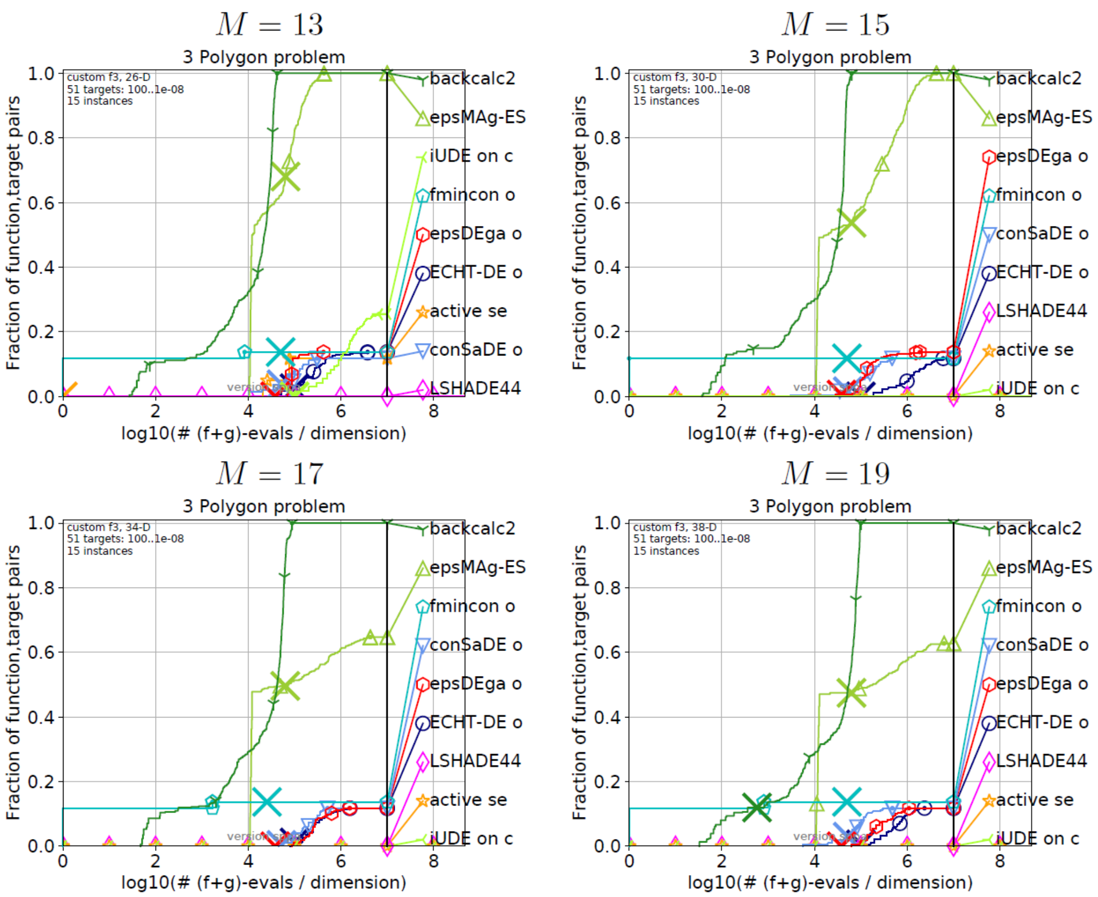


Figure 14: ECDF-plots of maximum area problem ($M = \text{nodes} - 1$).

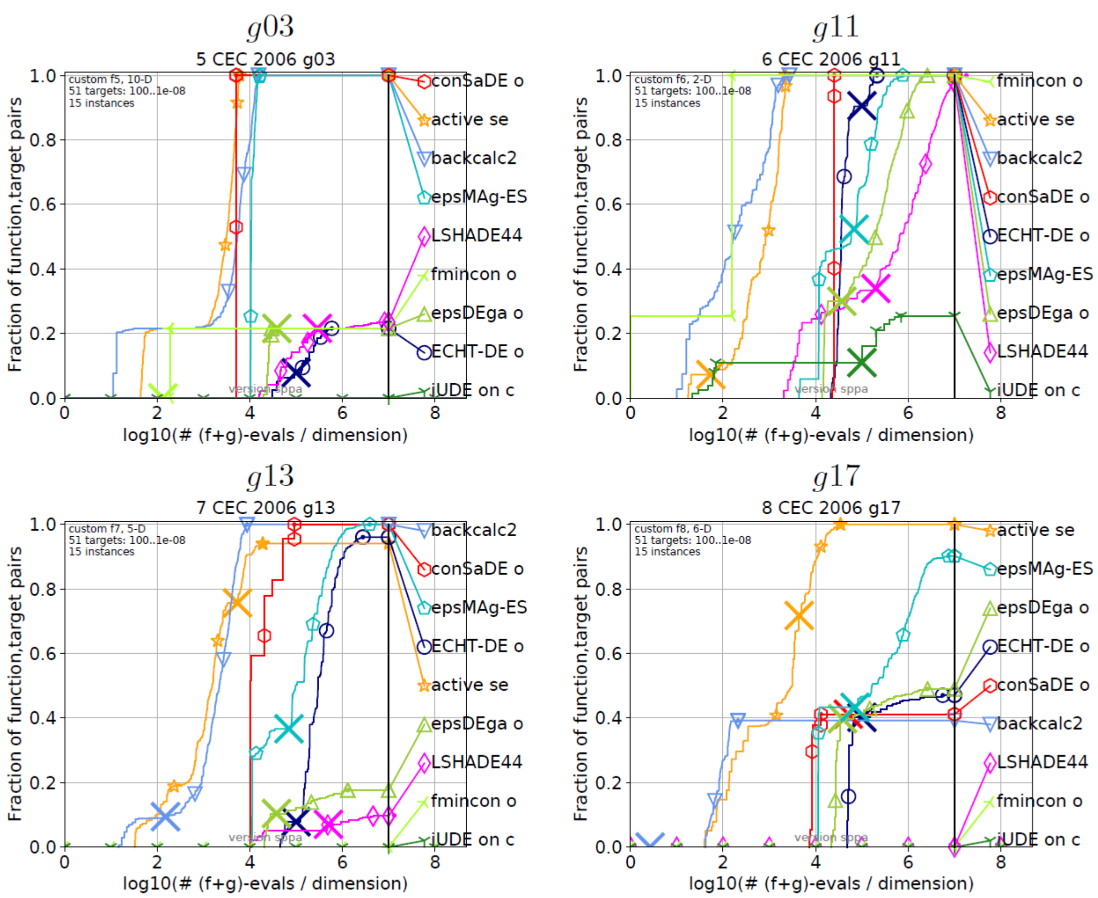


Figure 15: ECDF-plots of CEC06 equality constraint problems.

Inequality Constraints

- most constraint handling EAs are *NOT* interior point methods, i.e., they allow for *infeasible* solutions during the evolution process
- especially, the competitions at CEC and COCO BBOB allow for infeasible solutions

➥ there is a plethora of methods used in DE and PSO that are being considered only recently in Evolution Strategies (ES)

☞ the most successful will be considered below

- however, there is also an ES-specific approach by D.V. ARNOLD et al.[18] called "active covariance adaptation"
- idea is to incorporate even the *worst individuals'* direction vectors $\tilde{d}$ in the update of the covariance matrix $\mathbf{C}$ using *negative* weights w_l $(\forall l > \lambda/2)$

$$\mathbf{C} := \left(1 - \frac{1}{\tau_w}\right)\mathbf{C} + \frac{1}{\tau_w}\langle\tilde{\mathbf{d}}\tilde{\mathbf{d}}^{\mathrm{T}}\rangle_w \quad \text{where} \quad \langle\tilde{d}\tilde{d}^{\mathrm{T}}\rangle_w := \sum_{l=1}^{\lambda} w_l \tilde{\mathbf{d}}_{l;\lambda}\tilde{\mathbf{d}}_{l;\lambda}^{\mathrm{T}} \qquad (47)$$

[18] G. Jastrebski and D. Arnold. *Improving Evolution Strategies through Active Covariance Matrix Adaptation.* CEC'2006, pp. 2814–2821. DOI: 10.1109/CEC.2006.1688662

- this approach has been extended and used in an (1+1)-ES for constrained optimization[19]
- **Problem:** due to the negative weights in (47) $\mathbf{C}$ can become indefinite and $\sqrt{\mathbf{C}} \notin \mathbb{R}^{N_x \times N_x}$
- the novel $\mathbf{M}$ update (15) does *not* suffer from such problems, it solves the indefiniteness problem

Idea:

- for each constraint $g_k(\mathbf{x})$ (32) keep a fading record of $\mathbf{v}_k$-vectors that is updated in the case that kth constraint is violated for offspring $\tilde{\mathbf{x}}$

$$\forall k \in \{k | g_k(\tilde{\mathbf{x}}) > 0\}: \quad \mathbf{v}_k := \left(1 - \frac{1}{\tau_v}\right)\mathbf{v}_k + \frac{1}{\tau_v}\tilde{\mathbf{z}} \qquad (48)$$

- this learns the local normal direction of the constraint boundary from viewpoint of the isotropic $\mathbf{z}$ variation in the offspring generation loop

[19] D.V. Arnold & N. Hansen. *A (1+1)-CMA-ES for constrained optimisation.* GECCO'2012, pp. 297–304. DOI: 10.1145/2330163.2330207

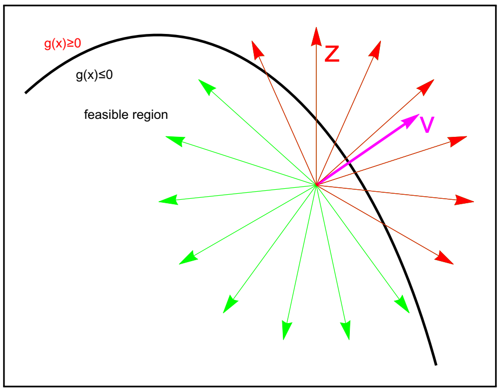


Figure 16: Those variations in the $(\mu/\mu_I, \lambda)$-MA-ES, Line M13 (Slide 30), which lead to infeasible offspring are cumulated according to (48). Thus, those parts of the infeasible $\mathbf{z}$ leading to mutations tangential to the feasibility border $g(\mathbf{x}) = 0$ are (approximately) averaged out, the normal part, being $\mathbf{v}$, remains.

- then, after selection, $\mathbf{M}$ is updated according to Line M13 (Slide 30) in the standard $(\mu/\mu_I, \lambda)$-MA-ES and in a second step
- the $\mathbf{v}_k$ normal directions of *all* violated $g(\tilde{\mathbf{x}}) \le 0$ constraints (within the actual generation) are incorporated in the $\mathbf{M}$ update ($\beta = \Theta(1/N_x)$)

$$\forall k \in \{k | g_k(\tilde{\mathbf{x}}) > 0\}: \quad \mathbf{M} := \mathbf{M} - \beta(\mathbf{M}\mathbf{v}_k)\mathbf{v}_k^{\mathrm{T}} \tag{49}$$

- the performance of the resulting MA-ES (pseudocode not displayed here[20]) have been compared to other approaches, especially to the one cited in footnote 19, see next slide

[20] Published in: P. Spettel & H.-G. Beyer. *A multi-recombinative active matrix adaptation evolution strategy for constrained optimization.* Soft Computing 23(16): 6847–6869. DOI: 10.1007/s00500-018-03736-z

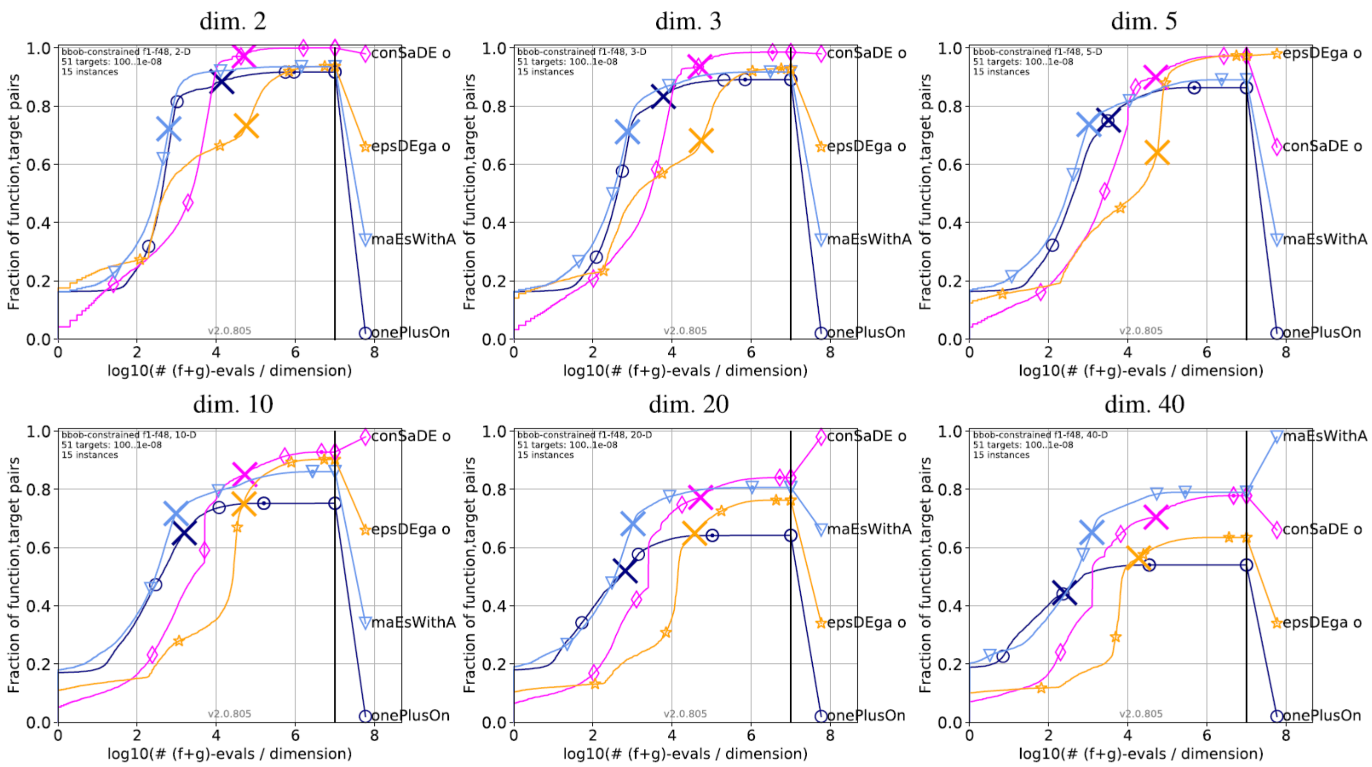


Figure 17: ECDF-plots of COCO BBOB for constraint problems comparing the active MA algorithm (maEsWithA) with the $(1+1)$-ES (Arnold & Hansen), conSaDE (Huang et al., 2006), and epsDEga (Takahama & Sakai, 2010).

How to Become Competitive[21]

General constrained minimization problem:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) \tag{50a}$$
$$\text{s.t.}\quad h_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, J \tag{50b}$$
$$g_k(\mathbf{x}) \leq 0, \qquad k = 1, \ldots, K \tag{50c}$$
$$\check{x}_n \leq (\mathbf{x})_n \leq \hat{x}_n, \quad n = 1, \ldots, N \tag{50d}$$

What are the ingredients for an MA-ES that is able to be on par or better than the currently best performing DE for constrained problems?

Résumé of an analysis of DE algorithms *and* the advantages of MA-ES:

1. always handle box-constraints (50d) first ⇒ KeepRange($\mathbf{x}$)
2. allow for infeasible solutions (if admissible as in CEC competitions)
3. use scheduled ϵ-*relaxed* lexicographic ordering of individuals
4. use infeasibility repair by gradient techniques "*now and then*"
5. in case of repair ❶ or ❹, calculate back to adapt the $\mathbf{M}$-matrix

[21] M. Hellwig & H.-G. Beyer. *A Matrix Adaptation Evolution Strategy for Constrained Real-Parameter Optimization*. CEC'2018, pp. 749–756. DOI: 10.1109/CEC.2018.8477950

Box Constraints

Different possiblities:

1. project onto the nearest box boundary
   - Advantage: provides a minimal repair (respects offspring locality)
   - Disadvantage: is biased towards corners of the box
2. reflect back into box
   - Advantage: no preference of certain box boundaries
   - Disadvantage: random point behavior (offspring locality violated)
   - however, it worked well in CEC competitions:

$$\text{KeepRange}(\mathbf{x})_n := \begin{cases} \check{x}_n + \left((\check{x}_n - x_n) - \left\lfloor \frac{\check{x}_n - x_n}{\hat{x}_n - \check{x}_n} \right\rfloor (\hat{x}_n - \check{x}_n) \right), & \text{if } x_n < \check{x}_n \\ \hat{x}_n - \left((x_n - \hat{x}_n) - \left\lfloor \frac{x_n - \hat{x}_n}{\hat{x}_n - \check{x}_n} \right\rfloor (\hat{x}_n - \check{x}_n) \right), & \text{if } x_n > \hat{x}_n \\ x_n, & \text{else} \end{cases} \tag{51}$$

ϵ-Relaxed Lexicographic Ordering[22]

Let

$$H_j(\mathbf{x}) := \begin{cases} |h_j(\mathbf{x})|, & \text{if } |h_j(\mathbf{x})| > \delta \\ 0, & \text{if } |h_j(\mathbf{x})| \leq \delta \end{cases} \tag{52}$$

and

$$G_k(\mathbf{x}) := \max(0, g_k(\mathbf{x})) \tag{53}$$

then infeasibility measure $\nu(\mathbf{x})$ is defined as

$$\nu(\mathbf{x}) := \sum_{j=1}^{J} H_j(\mathbf{x}) + \sum_{k=1}^{K} G_k(\mathbf{x}). \tag{54}$$

Given two individuals $\mathbf{x}_\alpha$ und $\mathbf{x}_\beta$ and the couple $(f_\alpha, \nu_\alpha) := (f(\mathbf{x}_\alpha), \nu(\mathbf{x}_\alpha))$, the ϵ-level lexicographic order relation $\preceq_\epsilon$ is defined (for f-minimization) as

$$\mathbf{x}_\alpha \preceq_\epsilon \mathbf{x}_\beta \;\Leftrightarrow\; \begin{cases} f_\alpha \leq f_\beta, & \text{if } (\nu_\alpha \leq \epsilon) \wedge (\nu_\beta \leq \epsilon), \\ f_\alpha \leq f_\beta, & \text{if } \nu_\alpha = \nu_\beta, \\ \nu_\alpha < \nu_\beta, & \text{otherwise.} \end{cases} \tag{55}$$

Note, in case of f-maximization, $f_\alpha \leq f_\beta$ is to be changed to $f_\alpha \geq f_\beta$.

[22] T. Takahama & S. Sakai. *Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites.* CEC'2006, pp. 308–315. DOI: 10.1109/CEC.2006.1688283

Design Principles for MA-ES on Constrained Problems ϵ-Relaxed Lexicographic Ordering

How to control ϵ

- finally $\epsilon \to 0$ must hold to ensure feasibility

$\Rightarrow$ reasonable to assume a generation number $g > T$ above which standard lexicographic ordering is used, i.e. $\epsilon = 0$

- currently, ϵ-decrease over the generations g is controlled by the *ad hoc* rule

$$\epsilon^{(g)} := \epsilon^{(0)} \left(1 - \frac{g}{T}\right)^{\gamma} \tag{56}$$

where

$$\epsilon^{(0)} := \frac{1}{\lfloor \theta_t \lambda \rfloor} \sum_{l=1}^{\lfloor \theta_t \lambda \rfloor} \nu\left(\mathbf{x}_{l;\lambda}^{(0)}\right) \tag{57}$$

- choice of T, $\theta_t \in (0, 1)$, and $\gamma \geq \gamma_{\min}$ by experimentation (in CEC 2018 competition: $T = 1000$, $\theta_t = 0.9$, and $\gamma_{\min} = 3$)

Gradient Based Repair

- applied *from time to time* (every Nth generation probabilistically) and *only approximately* (stopping after at most θ_r iteration steps)
- in case of an infeasible offspring $\tilde{\mathbf{x}} \Rightarrow$ repair by $\tilde{\mathbf{x}} := \tilde{\mathbf{x}} + \Delta\mathbf{x}$
- constraint vector: $\mathbf{c}(\mathbf{x}) := \big(h_1(\mathbf{x}), \ldots, h_J(\mathbf{x}), g_1(\mathbf{x}), \ldots, g_K(\mathbf{x})\big)^{\mathrm{T}}$
- Taylor: $c_m(\mathbf{x} + \Delta\mathbf{x}) = c_m(\mathbf{x}) + \nabla c_m^{\mathrm{T}} \Delta\mathbf{x} + \ldots \; (m = 1, \ldots, J + K)$
- demanding: $h_j(\mathbf{x}) + \nabla h_j^{\mathrm{T}} \Delta\mathbf{x} \overset{!}{=} 0$ and $g_k(\mathbf{x}) + \nabla g_k^{\mathrm{T}} \Delta\mathbf{x} \overset{!}{\leq} 0$
- neglecting higher order terms yields:
  $\nabla h_j^{\mathrm{T}} \Delta\mathbf{x} + h_j(\mathbf{x}) = 0$ and
  $\nabla g_k^{\mathrm{T}} \Delta\mathbf{x} + g_k(\mathbf{x}) \leq \nabla g_k^{\mathrm{T}} \Delta\mathbf{x} + \max(0, g_k(\mathbf{x})) = 0$
- collecting the gradients in a matrix $\mathbf{J}$ (the Jacobian), one obtains the linear system $\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{b}$, where

$$\mathbf{b}(\mathbf{x}) := \big(h_1(\mathbf{x}), \ldots, h_J(\mathbf{x}), \max(0, g_1(\mathbf{x})), \ldots, \max(0, g_K(\mathbf{x}))\big)^{\mathrm{T}} \tag{58}$$

- using pseudoinverse $\mathbf{J}^{\dagger}$, an offspring repair update step reads

$$\tilde{\mathbf{x}} := \tilde{\mathbf{x}} - \mathbf{J}^{\dagger}(\tilde{\mathbf{x}})\mathbf{b}(\tilde{\mathbf{x}}) \tag{59}$$

  which can be executed θ_r times in a row if need be

MA-ES Specific Step: $\tilde{\mathbf{z}}$ Back Calculation

- as in the case of equality constraint repair (Slide 45ff), back calculation is (often) benefical after a Repair and/or KeepRange step
- let $\tilde{\mathbf{x}}_l$ the *repaired* offspring l, then ($\mathbf{x}$ being the parental state)

$$\tilde{\mathbf{d}}_l := \frac{1}{\sigma}(\tilde{\mathbf{x}}_l - \mathbf{x}) \tag{60}$$

  and

$$\tilde{\mathbf{z}}_l := \mathbf{M}_{\mathrm{inv}} \tilde{\mathbf{d}}_l, \tag{61}$$

  where $\mathbf{M}_{\mathrm{inv}}$ can be:

  1. the pseudoinverse $\mathbf{M}^{\dagger}$ of $\mathbf{M}$[23] or
  2. evolved using the update (46)[24]

[23] Used in our publication mentioned in Footnote 21.

[24] This approach needs further investigations regarding the general problem (50).

On the Influence of Different Algorithmic Ingredients

ϵMAg-ES	$N = 10$			$N = 100$		
		Ranking			Ranking	
+/=/−	Median	Mean	Total	Median	Mean	Total
ϵMA-ES	7/19/2	7/17/4	7/18/3	5/12/11	7/11/10	5/13/10
ϵMAg-ES w/o	8/17/2	10/15/3	10/15/3	10/8/10	13/6/9	11/8/9
ϵMAg-ES nl	12/14/2	14/12/2	14/12/2	5/15/8	7/14/7	5/16/7
ϵSAg-ES	18/9/1	20/7/1	20/7/1	18/8/2	18/8/2	17/10/1
*lex*MAg-ES	6/20/2	7/18/3	7/18/3	9/10/9	10/9/9	9/11/8
*lex*MA-ES	8/16/4	8/14/6	8/14/6	14/7/7	14/7/7	13/9/6

Figure 18: The influence of switching off different algorithmic ingredients in the ϵMAg-ES on the performance using the constrained CEC2017 benchmark. Missing "g": no gradient based repair; "w/o": no $\mathbf{z}$ back calculation; "nl": no σ-limitation; "SA": $\mathbf{M} \equiv \mathbf{I}$; "*lex*": $\epsilon \equiv 0$.
"$+/=/-$": number of problems where ϵMAg-ES is significantly "better than / on par with / worse than" the downgraded versions.

Summary

The C in CMA-ES can be removed yielding MA-ES

- this simplifies the ES algorithm and provides deeper insights:
  - $\sqrt{\mathbf{C}}$ operation is no longer needed (no problems with negative eigenvalues)
  - one may also remove the $\mathbf{d}$ evolution path
- provides a simple interpretation of the ES working principles in that the evolution of the $\mathbf{M}$-matrix is governed by the selection-caused deviation from the isotropically generated random $\mathbf{z}$ vectors

⇒ MA-ES seeks to transform the optimization problem locally into a sphere model

These results/findings gave and give rise to new algorithm designs:

- approximating the **M**-matrix with a few cumulated vectors allows for limited memory LM-MA-ES that works for search space dimensionalities of quite a few thousands (and even more)[25]
- the MA evolution idea can be transferred to constrained optimization:
  - infeasible solutions can be easily used to improve the **M**-matrix
  - also repaired solutions can be used in a **z** back calculation step to improve the **M**-matrix
  - the "inverse" **M**-matrix can also be *evolved* (i.e., w/o explicit inversion operations)
  - using similar techniques as have been used in DE (Differential Evolution), one can easily design ESs that are among the best performing algorithms

Algorithm design based on MA-ES has just begun.
You are invited to enter this field!

Thank You For Your Attention!

[25] Unlike most CMA-ES versions proposed for higher search space dimensionalities no assumptions regarding diagonal or block structure are needed.

Related Publications I

H.-G. Beyer.
Evolution Strategies.
Scholarpedia, 2(8):1965, 2007.

H.-G. Beyer and H.-P. Schwefel.
Evolution Strategies: A Comprehensive Introduction.
Natural Computing, 1(1):3–52, 2002.

N. Hansen, S.D. Müller, and P. Koumoutsakos.
Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES).
Evolutionary Computation, 11(1):1–18, 2003.

H.-G. Beyer and B. Sendhoff.
Simplify Your Covariance Matrix Adaptation Evolution Strategy.
IEEE Transactions on Evolutionary Computation, 21(5):746–759, 2017.
DOI: 10.1109/TEVC.2017.2680320.

H.-G. Beyer and B. Sendhoff.
Covariance Matrix Adaptation Revisited – the CMSA Evolution Strategy.
In G. Rudolph et al., editor, *Parallel Problem Solving from Nature 10*, pages 123–132, Berlin, 2008. Springer.
DOI: 10.1007/978-3-540-87700-4_13.

H.-G. Beyer, J. Edler, M. Herdy, I. Santibanez-Koref, M. Olhofer, G. Rudolph, W. Sachs, S. Schliewe, H. Seitz, and I. Tesari.
VDI-Richtlinie 6224, Blatt 1: Bionische Optimierung – Evolutionäre Algorithmen in der Anwendung.
In *VDI-Handbuch Bionik*. Verein Deutscher Ingenieure, Düsseldorf, 2012.

Related Publications II

H.-G. Beyer and D.V Arnold.
Qualms Regarding the Optimality of Cumulative Path Length Control in CSA/CMA-Evolution Strategies.
Evolutionary Computation, 11(1):19–28, 2003.
DOI: 10.1162/EVCO_a_00261.

D.V. Arnold and H.-G. Beyer.
Performance Analysis of Evolutionary Optimization With Cumulative Step Length Adaptation.
IEEE Transactions on Automatic Control, 49(4):617–622, 2004.

N. Hansen and A. Ostermeier.
Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation.
In *Proceedings of 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC '96)*, pages 312–317. IEEE Press, NY, 1996.

O. Krause, D.R. Arbones, and C. Igel.
CMA-ES with optimal covariance Update and Storage Complexity.
In *Proc. Adv. Neural Inf. Process. Syst.*, pages 370–378, Barcelona, Spain, 2016.

I. Loshchilov, T. Glasmachers, and H.-G. Beyer.
Large Scale Black-box Optimization by Limited-Memory Matrix Adaptation.
IEEE Transactions on Evolutionary Computation, 23(2):353–358, 2019.
DOI: 10.1109/TEVC.2018.2855049.

N. Hansen.
The CMA Evolution Strategy: A Comparing Review.
In J.A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, pages 75–102. Springer, 2006.

Related Publications III

D.V. Arnold.
Reconsidering constraint release for active-set evolution strategies.
In *GECCO'17: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 665–672, New York, 2017. ACM.
DOI: 10.1145/3071178.3071294.

P. Spettel, H.-G. Beyer, and M. Hellwig.
A Covariance Matrix Self-Adaptation Evolution Strategy for Optimization under Linear Constraints.
IEEE Transactions on Evolutionary Computation, 23(3):514–524, 2019.
https://doi.org/10.1109/TEVC.2018.2871944.

P. Spettel and H.-G. Beyer.
Matrix Adaptation Evolution Strategies for Optimization Under Nonlinear Equality Constraints.
Swarm and Evolutionary Computation, 2019.
DOI: 10.1016/j.swevo.2020.100653.

G.A. Jastrebski and D.V. Arnold.
Improving Evolution Strategies through Active Covariance Matrix Adaptation.
In *Proceedings of the CEC'06 Conference*, pages 2814–2821, Piscataway, NJ, 2006. IEEE.

D.V. Arnold and N. Hansen.
A (1+1)-CMA-ES for constrained optimisation.
In *GECCO'12: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 297–304, New York, 2012. ACM.

P. Spettel and H.-G. Beyer.
A multi-recombinative active matrix adaptation evolution strategy for constrained optimization.
Soft Computing, 23(16):6847–6869, 2019.
https://doi.org/10.1007/s00500-018-03736-z.

Related Publications IV

M. Hellwig and H.-G. Beyer.
A Matrix Adaptation Evolution Strategy for Constrained Real-Parameter Optimization.
In *Proceedings of the WCCI'18 Conference*, pages 749–756, Piscataway, NJ, 2018. IEEE Press.
DOI: 10.1109/CEC.2018.8477950.

T. Takahama and S. Sakai.
Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites.
In *CEC 2006 IEEE Congress on Evolutionary Computation*, pages 308–315, 2006.
DOI: 10.1109/CEC.2006.1688283.