Tutorial: CMA-ES — Evolution Strategies and Covariance Matrix Adaptation

Anne Auger & Nikolaus Hansen

INRIA Research Centre Saclay – Île-de-France
Project team TAO
University Paris-Sud, LRI (UMR 8623), Bat. 490
91405 ORSAY Cedex, France

GECCO'11, July 12–16, 2011, Dublin, Ireland.
get the slides: google "Nikolaus Hansen"... under Publications click Invited talks, tutorials...

Content

Problem Statement

Continuous Domain Search/Optimization

- Task: **minimize** an **objective function** (*fitness* function, *loss* function) in continuous domain

$$f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}, \qquad \boldsymbol{x} \mapsto f(\boldsymbol{x})$$

- **Black Box** scenario (direct search scenario)

  - gradients are not available or not useful
  - problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding
- Search costs: number of function evaluations

Problem Statement

Continuous Domain Search/Optimization

- Goal
  - fast convergence to the global optimum

    . . . or to a robust solution $\boldsymbol{x}$
  - solution $\boldsymbol{x}$ with **small function value** $f(\boldsymbol{x})$ with **least search cost**

    there are two conflicting objectives
- Typical Examples
  - shape optimization (e.g. using CFD) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images
- Problems
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Objective Function Properties

We assume $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ to be *non-linear, non-separable* and to have at least moderate dimensionality, say $n \not\ll 10$.
Additionally, f can be

- non-convex
- multimodal

  there are possibly many local optima
- non-smooth

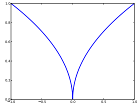
  derivatives do not exist
- discontinuous
- ill-conditioned
- noisy
- ...

Goal : cope with any of these function properties
they are related to real-world problems

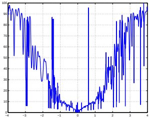
What Makes a Function Difficult to Solve?

Why stochastic search?

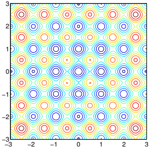
- non-linear, non-quadratic, non-convex
  on linear and quadratic functions much better search policies are available



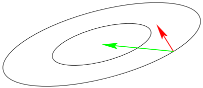
- ruggedness
  non-smooth, discontinuous, multimodal, and/or noisy function



- dimensionality (size of search space)
  (considerably) larger than three



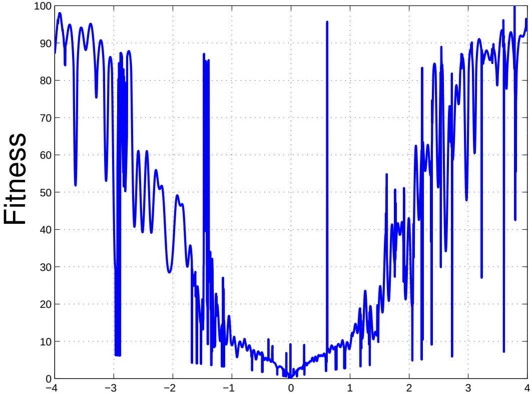
- non-separability
  dependencies between the objective variables
- ill-conditioning



gradient direction Newton direction

Ruggedness

non-smooth, discontinuous, multimodal, and/or noisy



cut from a 5-D example, (easily) solvable with evolution strategies

Curse of Dimensionality

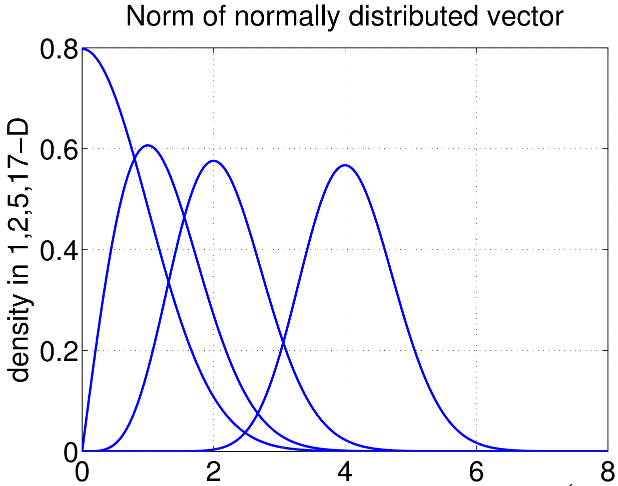
The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the **rapid increase in volume** associated with adding extra dimensions to a (mathematical) space.

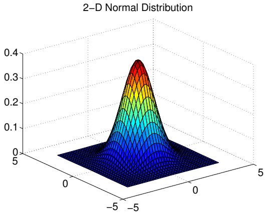
Example: Consider placing 100 points onto a real interval, say $[0,1]$. To get **similar coverage**, in terms of distance between adjacent points, of the 10-dimensional space $[0,1]^{10}$ would require $100^{10}=10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

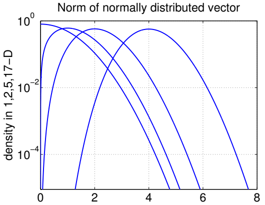
Remark: **distance measures** break down in higher dimensionalities (the central limit theorem kicks in)

Consequently, a **search policy** (e.g. exhaustive search) that is valuable in small dimensions **might be useless** in moderate or large dimensional search spaces.

Effect of Dimensionality: Example







$\|\mathcal{N}(\mathbf{0},\mathbf{I}) - \mathcal{N}(\mathbf{0},\mathbf{I})\|/\sqrt{2} \sim \|\mathcal{N}(\mathbf{0},\mathbf{I})\| \longrightarrow \mathcal{N}\left(\sqrt{n-1/2}, \mathbf{1/2}\right)$,

with modal value: $\sqrt{n-1}$

Separable Problems

Definition (Separable Problem)

A function f is separable if

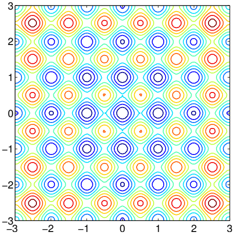
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

$\Rightarrow$ it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example: Additively decomposable functions

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i)$$

Rastrigin function



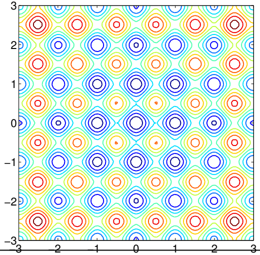
Non-Separable Problems

Building a non-separable problem from a separable one [1,2]

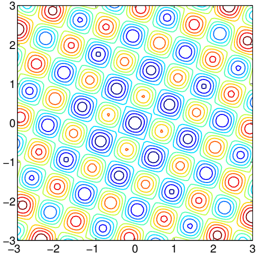
Rotating the coordinate system

- $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ separable
- $f : \boldsymbol{x} \mapsto f(\mathbf{R}\boldsymbol{x})$ **non-separable**

$\mathbf{R}$ rotation matrix



$$\xrightarrow{\mathbf{R}}$$



[1] Hansen, Ostermeier, Gawelczyk (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. Sixth ICGA, pp. 57-64, Morgan Kaufmann

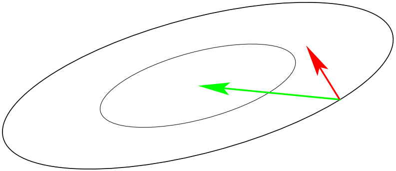
[2] Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function
$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^T\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^*) = \frac{1}{2}\sum_i h_{i,i}\,x_i^2 + \frac{1}{2}\sum_{i\neq j} h_{i,j}\,x_ix_j$

$\boldsymbol{H}$ is Hessian matrix of f and symmetric positive definite



gradient direction $-f'(\boldsymbol{x})^{\mathrm{T}}$

Newton direction $-\boldsymbol{H}^{-1}f'(\boldsymbol{x})^{\mathrm{T}}$

Ill-conditioning means **squeezed level sets** (high curvature). Condition number equals nine here. Condition numbers up to 10^{10} are not unusual in real world problems.

If $\boldsymbol{H} \approx \mathbf{I}$ (small condition number of $\boldsymbol{H}$) first order information (e.g. the gradient) is sufficient. Otherwise **second order information** (estimation of $\boldsymbol{H}^{-1}$) **is necessary.**

What Makes a Function Difficult to Solve?

...and what can be done

The Problem	The Approach in ESs and continuous EDAs
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric
Ruggedness	**non-local** policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, **population-based** method
	recombination operator serves as repair mechanism
	restarts

...metaphors

Metaphors

Evolutionary Computation		Optimization
individual, offspring, parent	$\longleftrightarrow$	candidate solution
		decision variables
		design variables
		object variables
population	$\longleftrightarrow$	set of candidate solutions
fitness function	$\longleftrightarrow$	objective function
		loss function
		cost function
		error function
generation	$\longleftrightarrow$	iteration

. . . methods: ESs

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. **Sample distribution** $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. **Evaluate** $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ **on** f
3. **Update parameters** $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via **operators on a population**, in particular, selection, recombination and mutation

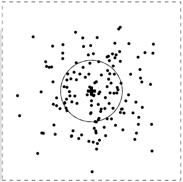
Natural template for *Estimation of Distribution Algorithms*

Evolution Strategies

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0},\mathbf{C}) \qquad \text{for } i = 1,\ldots,\lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n\times n}$



where

- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$ determines the **shape** of the distribution ellipsoid

here, all new points are sampled with the same parameters

The question remains how to update $\boldsymbol{m}$, $\mathbf{C}$, and σ.

Why Normal Distributions?

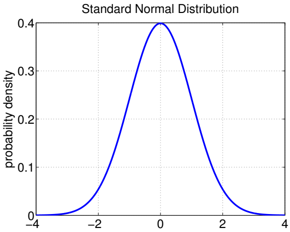
1. widely observed in nature, for example as phenotypic traits
2. only stable distribution with finite variance

   stable means the sum of normal variates is again normal,
   helpful in **design and analysis** of algorithms
   connection to central limit theorem
3. most convenient way to generate **isotropic** search points

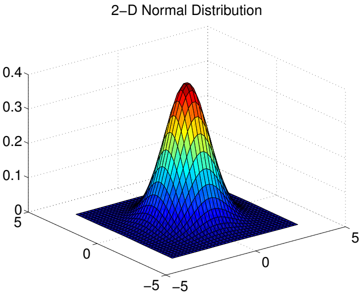
   the isotropic distribution does **not favor any direction**
   (unfoundedly), supports rotational invariance
4. maximum entropy distribution with finite variance

   the least possible assumptions on f in the distribution shape

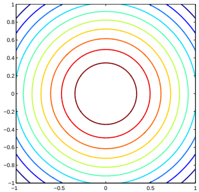
Normal Distribution



probability density of the 1-D standard normal distribution



probability density of a 2-D normal distribution

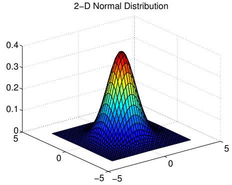


The Multi-Variate (n-Dimensional) Normal Distribution

Any multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$ is uniquely determined by its mean value $\boldsymbol{m} \in \mathbb{R}^n$ and its symmetric positive definite $n \times n$ covariance matrix $\mathbf{C}$.

The **mean** value $\boldsymbol{m}$

- determines the displacement (translation)
- value with the largest density (modal value)
- the distribution is symmetric about the distribution mean

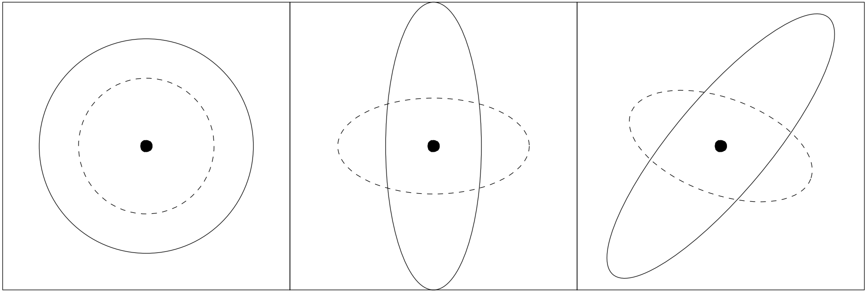


The **covariance matrix** $\mathbf{C}$

- determines the shape
- **geometrical interpretation**: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}} \mathbf{C}^{-1} (\boldsymbol{x} - \boldsymbol{m}) = 1\}$

... any **covariance matrix** can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}} \mathbf{C}^{-1} (\boldsymbol{x} - \boldsymbol{m}) = 1\}$

Lines of Equal Density



$\mathcal{N}\left(\boldsymbol{m}, \sigma^2\mathbf{I}\right) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$
one degree of freedom σ
components are independent standard normally distributed

$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$
n **degrees of freedom**
components are independent, scaled

$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
$(n^2 + n)/2$ **degrees of freedom**
components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ holds for all $\mathbf{A}$.

... ESs

Evolution Strategies

Terminology

Let μ: # of parents, λ: # of offspring

Plus (elitist) and comma (non-elitist) selection

$(\mu + \lambda)$-ES: selection in $\{\text{parents}\} \cup \{\text{offspring}\}$
(μ, λ)-ES: selection in $\{\text{offspring}\}$

$(1 + 1)$-ES

Sample one offspring from parent $\boldsymbol{m}$

$$\boldsymbol{x} = \boldsymbol{m} + \sigma \mathcal{N}(\boldsymbol{0}, \mathbf{C})$$

If $\boldsymbol{x}$ better than $\boldsymbol{m}$ select

$$\boldsymbol{m} \leftarrow \boldsymbol{x}$$

... why?

The $(\mu/\mu, \lambda)$-ES

Non-elitist selection and intermediate (weighted) recombination

Given the i-th solution point $\boldsymbol{x}_i = \boldsymbol{m} + \sigma \underbrace{\mathcal{N}_i(\mathbf{0}, \mathbf{C})}_{=:\, \boldsymbol{y}_i} = \boldsymbol{m} + \sigma\, \boldsymbol{y}_i$

Let $\boldsymbol{x}_{i:\lambda}$ the ***i*-th ranked** solution point, such that $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$. The new mean reads

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i:\lambda} \;=\; \boldsymbol{m} + \sigma \underbrace{\sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda}}_{=:\, \boldsymbol{y}_w}$$

where

$$w_1 \geq \cdots \geq w_\mu > 0, \quad \textstyle\sum_{i=1}^{\mu} w_i = 1, \quad \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} =: \mu_w \approx \frac{\lambda}{4}$$

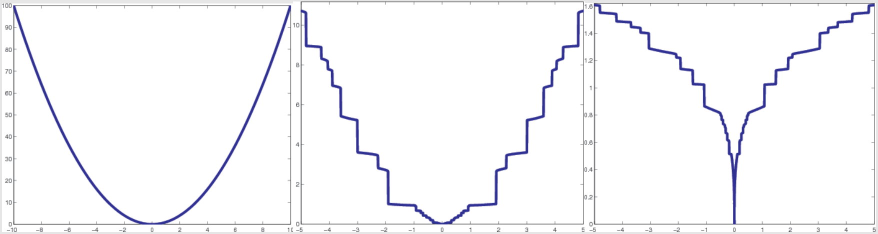
The best μ points are selected **from the new solutions** (non-elitistic) and **weighted intermediate recombination** is applied.

Invariance Under Monotonically Increasing Functions

Rank-based algorithms

Update of all parameters uses only the ranks

$$f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \ldots \leq f(x_{\lambda:\lambda})$$



$$g(f(x_{1:\lambda})) \leq g(f(x_{2:\lambda})) \leq \ldots \leq g(f(x_{\lambda:\lambda})) \quad \forall g$$

g is strictly monotonically increasing
g preserves ranks

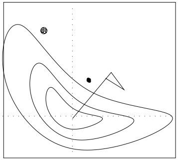
3

[3] Whitley 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, ICGA

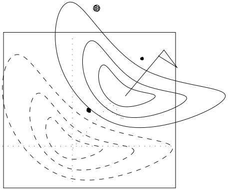
Basic Invariance in Search Space

- translation invariance

is true for most optimization algorithms



$$f(\boldsymbol{x}) \leftrightarrow f(\boldsymbol{x}-\boldsymbol{a})$$



Identical behavior on f and $f_{\boldsymbol{a}}$

$$\begin{aligned} f: &\quad \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 \\ f_{\boldsymbol{a}}: &\quad \boldsymbol{x} \mapsto f(\boldsymbol{x}-\boldsymbol{a}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 + \boldsymbol{a} \end{aligned}$$

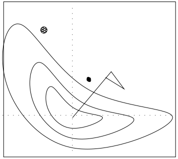
No difference can be observed w.r.t. the argument of f

Rotational Invariance in Search Space

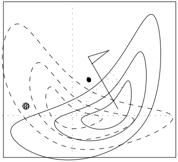
- invariance to orthogonal (rigid) transformations $\mathbf{R}$, where $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$

e.g. true for simple evolution strategies

recombination operators might jeopardize rotational invariance



$f(\boldsymbol{x}) \leftrightarrow f(\mathbf{R}\boldsymbol{x})$



Identical behavior on f and $f_{\mathbf{R}}$

$$\begin{aligned} f: &\quad \boldsymbol{x} \mapsto f(\boldsymbol{x}), &\quad \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 \\ f_{\mathbf{R}}: &\quad \boldsymbol{x} \mapsto f(\mathbf{R}\boldsymbol{x}), &\quad \boldsymbol{x}^{(t=0)} &= \mathbf{R}^{-1}(\boldsymbol{x}_0) \end{aligned}$$

45

No difference can be observed w.r.t. the argument of f

[4] Salomon 1996. "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

[5] Hansen 2000. Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies. *Parallel Problem Solving from Nature PPSN VI*

Invariance

Impact

The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.

— Albert Einstein

- Empirical performance results, for example
  - from benchmark functions
  - from solved real world problems

  are only useful if they do **generalize** to other problems

- **Invariance** is a strong **non-empirical** statement about generalization

  generalizing (identical) performance from a single function to a whole class of functions

consequently, invariance is important for the evaluation of search algorithms

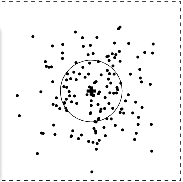
Evolution Strategies

Recalling

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma\,\mathcal{N}_i(\boldsymbol{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

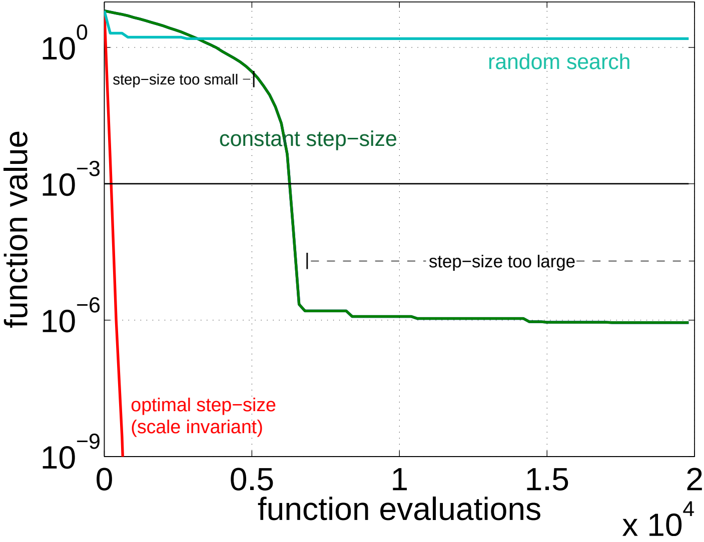


where

- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution and $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

The remaining question is how to update σ and $\mathbf{C}$.

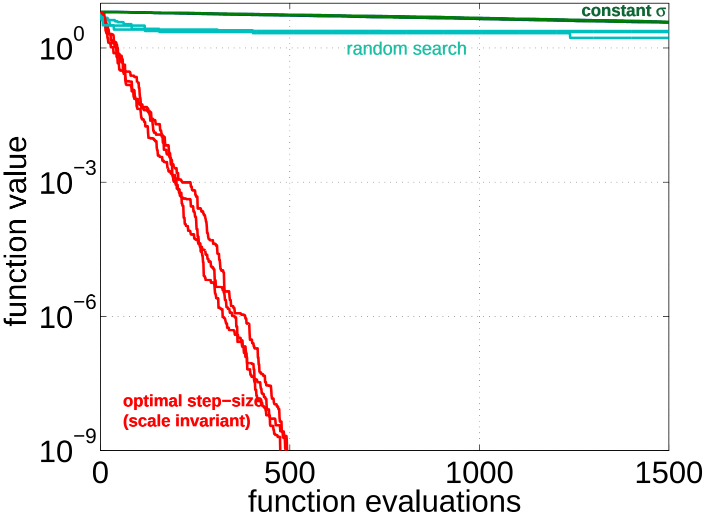
Why Step-Size Control?



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

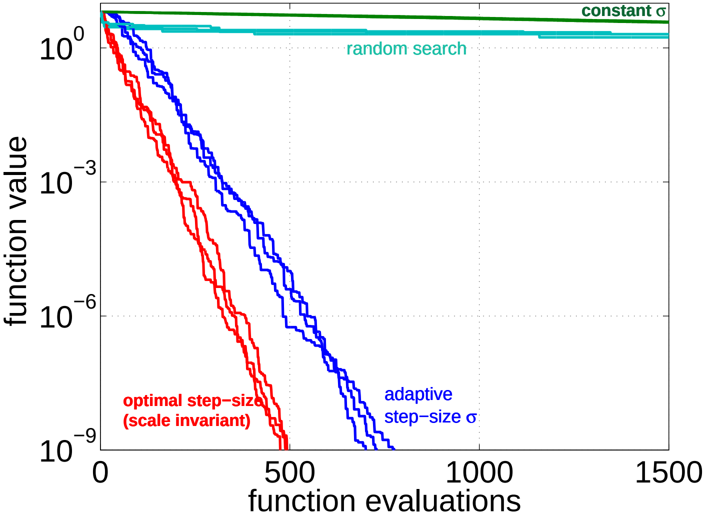
Why Step-Size Control?



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

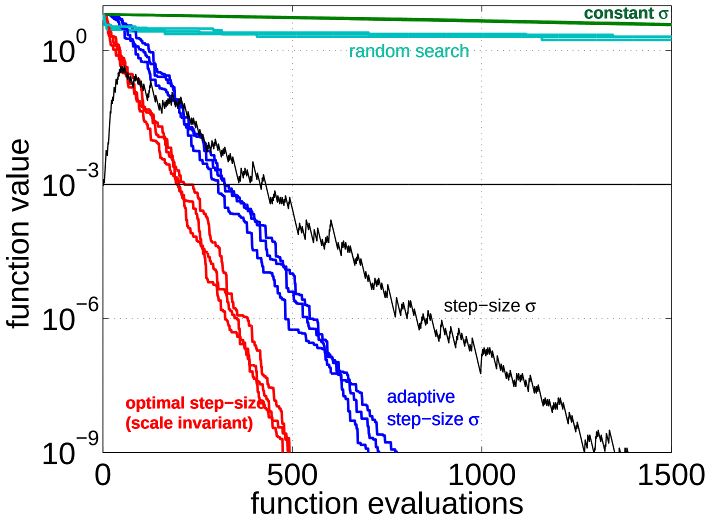
Why Step-Size Control?



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

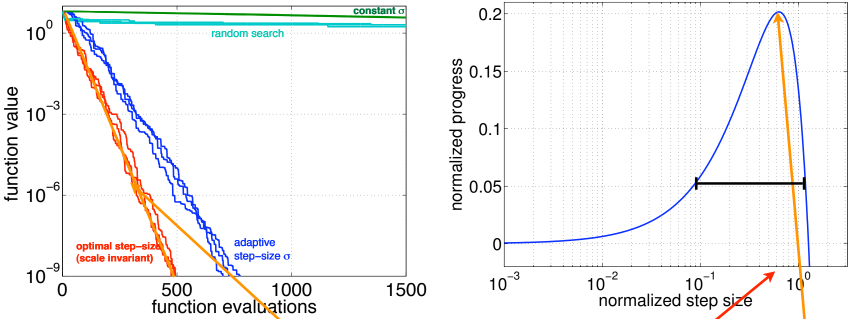
Why Step-Size Control?



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

Why Step-Size Control?



$\sigma \leftarrow \sigma^*_{\mathrm{opt}} \|\mathrm{parent}\|$

$-\dfrac{\varphi^*}{n}$

σ^*_{opt}

φ^*

evolution window refers to the step-size interval (⊢—⊣) where reasonable performance is observed

Methods for Step-Size Control

- $1/5$-th success rule[a][b], often applied with "+"-selection

  increase step-size if more than 20% of the new solutions are successful, decrease otherwise

- σ-self-adaptation[c], applied with ","-selection

  mutation is applied to the step-size and the better, according to the objective function value, is selected

  simplified "global" self-adaptation

- path length control[d] (Cumulative Step-size Adaptation, CSA)[e]

  self-adaptation derandomized and non-localized

---

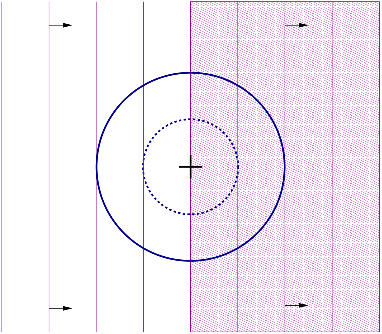
[a] Rechenberg 1973, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog

[b] Schumer and Steiglitz 1968. Adaptive step size random search. *IEEE TAC*

[c] Schwefel 1981, *Numerical Optimization of Computer Models*, Wiley

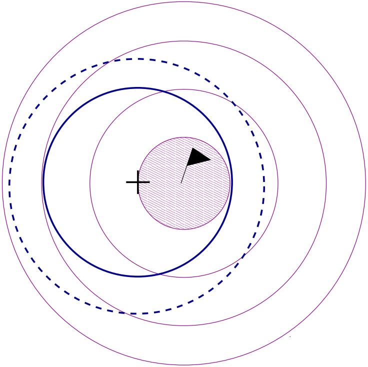
[d] Hansen & Ostermeier 2001, Completely Derandomized Self-Adaptation in Evolution Strategies, *Evol. Comput.* 9(2)

One-fifth success rule



↓

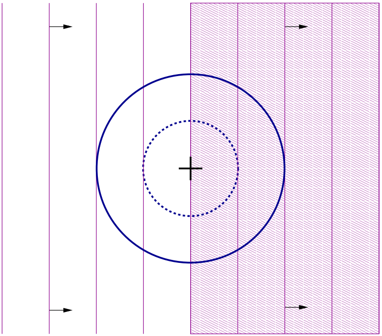
increase σ

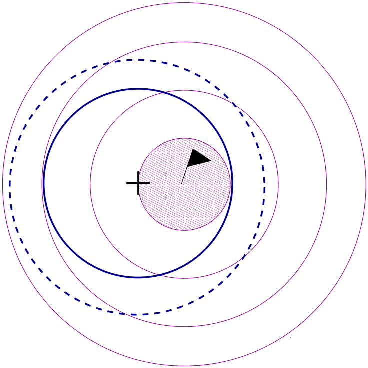


↓

decrease σ

One-fifth success rule





Probability of success (p_s)

Probability of success (p_s)

$1/2$

$1/5$

“too small”

One-fifth success rule

p_s: # of successful offspring / # offspring (per generation)

$$\sigma \leftarrow \sigma \times \exp\left(\frac{1}{3} \times \frac{p_s - p_{\text{target}}}{1 - p_{\text{target}}}\right)$$

Increase σ if $p_s > p_{\text{target}}$
Decrease σ if $p_s < p_{\text{target}}$

$(1+1)$-ES

$p_{target} = 1/5$

IF *offspring better parent*

$p_s = 1,\ \sigma \leftarrow \sigma \times \exp(1/3)$

ELSE

$p_s = 0,\ \sigma \leftarrow \sigma / \exp(1/3)^{1/4}$

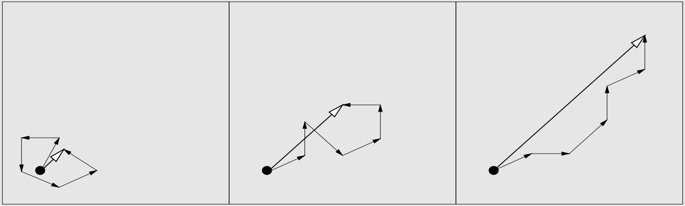
Path Length Control (CSA)

The Concept of Cumulative Step-Size Adaptation

$$\begin{aligned} x_i &= m + \sigma\, y_i \\ m &\leftarrow m + \sigma y_w \end{aligned}$$

Measure the length of the *evolution path*

the pathway of the mean vector m in the generation sequence



$\Downarrow$ decrease σ $\Downarrow$ increase σ

loosely speaking steps are

- perpendicular under random selection (in expectation)
- perpendicular in the desired situation (to be most efficient)

Path Length Control (CSA)

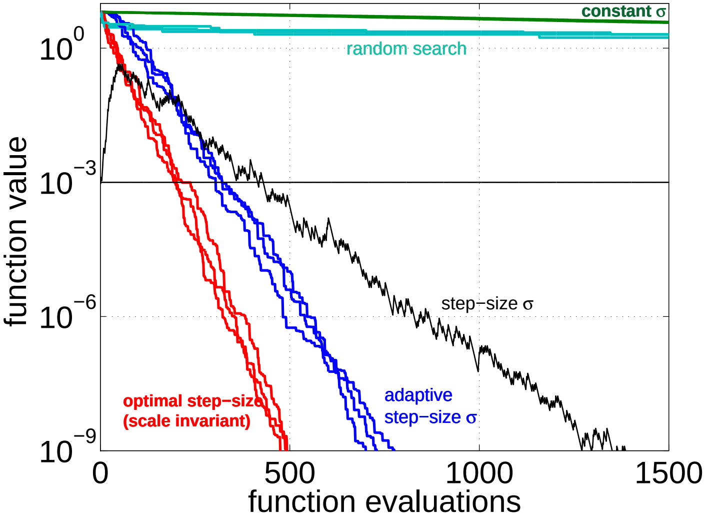
The Equations

Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \quad \text{where } \boldsymbol{y}_w = \textstyle\sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \qquad \text{update mean}$$

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \underbrace{\sqrt{1 - (1 - c_\sigma)^2}}_{\text{accounts for } 1-c_\sigma} \underbrace{\sqrt{\mu_w}}_{\text{accounts for } w_i} \boldsymbol{y}_w$$

$$\sigma \leftarrow \sigma \times \underbrace{\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)}_{>1 \iff \|\boldsymbol{p}_\sigma\| \text{ is greater than its expectation}} \qquad \text{update step-size}$$



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

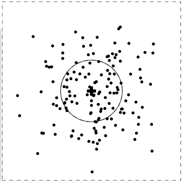
Evolution Strategies

Recalling

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$



where

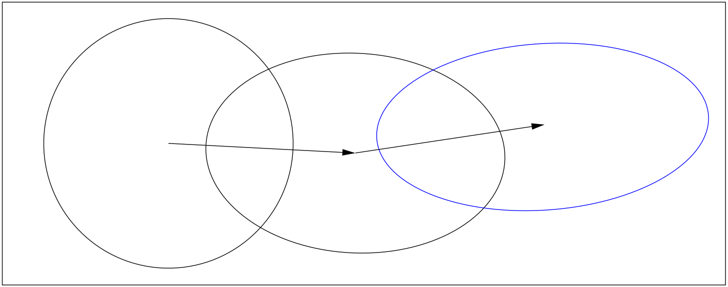
- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

The remaining question is how to update $\mathbf{C}$.

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



new distribution,

$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

the ruling principle: the adaptation **increases the likelihood of successful steps**, $\boldsymbol{y}_w$, to appear again

another viewpoint: the adaptation **follows a natural gradient** approximation of the expected fitness

. . . equations

Covariance Matrix Adaptation

Rank-One Update

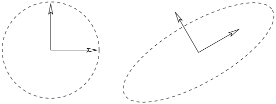
Initialize $\boldsymbol{m} \in \mathbb{R}^n$, and $\mathbf{C} = \mathbf{I}$, set $\sigma = 1$, learning rate $c_{\text{cov}} \approx 2/n^2$
While not terminate

$$
\begin{aligned}
\boldsymbol{x}_i &= \boldsymbol{m} + \sigma\, \boldsymbol{y}_i, \qquad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\
\boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \qquad \text{where } \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \\
\mathbf{C} &\leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \mu_w \underbrace{\boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}}_{\text{rank-one}} \qquad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} \geq 1
\end{aligned}
$$

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \mu_w \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$$

covariance matrix adaptation

- learns all **pairwise dependencies** between variables
  off-diagonal entries in the covariance matrix reflect the dependencies
- conducts a **principle component analysis** (PCA) of steps $\boldsymbol{y}_w$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid, rotational invariant
- learns a new, **rotated problem representation** and a **new metric** (Mahalanobis)



  components are independent (only) in the new representation
  rotational invariant
- approximates the **inverse Hessian** on quadratic functions
  overwhelming empirical evidence, proof is in progress
- is **entirely independent** of the given coordinate system
  for $\mu = 1$: natural gradient ascent on $\mathcal{N}$

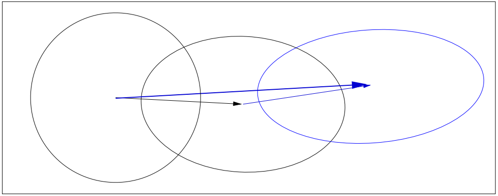
. . . cumulation, rank-μ

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the search path the strategy takes over a number of generation steps. It can be expressed as a sum of consecutive *steps* of the mean $\boldsymbol{m}$.



An exponentially weighted sum of steps $\boldsymbol{y}_w$ is used

$$\boldsymbol{p}_\mathrm{c} \propto \sum_{i=0}^{g} \underbrace{(1-c_\mathbf{c})^{g-i}}_{\substack{\text{exponentially}\\ \text{fading weights}}} \boldsymbol{y}_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$\boldsymbol{p}_\mathrm{c} \leftarrow \underbrace{(1-c_\mathbf{c})}_{\text{decay factor}} \boldsymbol{p}_\mathrm{c} + \underbrace{\sqrt{1-(1-c_\mathbf{c})^2}\sqrt{\mu_w}}_{\text{normalization factor}} \underbrace{\boldsymbol{y}_w}_{\text{input } = \frac{\boldsymbol{m}-\boldsymbol{m}_\text{old}}{\sigma}}$$

where $\mu_w = \frac{1}{\sum {w_i}^2}$, $c_\mathbf{c} \ll 1$. History information is accumulated in the evolution path.

"Cumulation" is a widely used technique and also know as

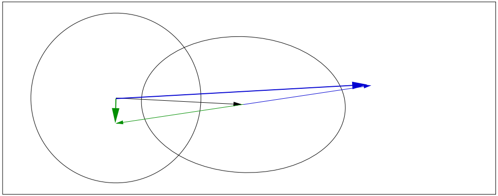
- *exponential smoothing* in time series, forecasting
- exponentially weighted *mooving average*
- *iterate averaging* in stochastic approximation
- *momentum* in the back-propagation algorithm for ANNs
- ...

... why?

Cumulation

Utilizing the Evolution Path

We used $\boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}} = -\boldsymbol{y}_w(-\boldsymbol{y}_w)^{\mathrm{T}}$ the sign of $\boldsymbol{y}_w$ is lost.



The sign information is (re-)introduced by using the *evolution path*.

$$\boldsymbol{p}_\mathrm{c} \leftarrow \underbrace{(1 - c_\mathbf{c})}_{\text{decay factor}} \boldsymbol{p}_\mathrm{c} + \underbrace{\sqrt{1 - (1 - c_\mathbf{c})^2} \sqrt{\mu_w}}_{\text{normalization factor}} \boldsymbol{y}_w$$

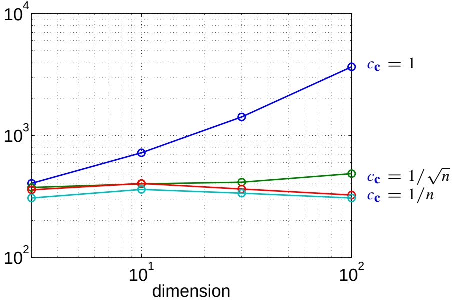
$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}} \underbrace{\boldsymbol{p}_\mathrm{c} \boldsymbol{p}_\mathrm{c}^{\mathrm{T}}}_{\text{rank-one}}$$

where $\mu_w = \frac{1}{\sum {w_i}^2}$, $c_\mathbf{c} \ll 1$ such that $1/c_\mathbf{c}$ is the "backward time horizon".

Using an **evolution path** for the **rank-one update** of the covariance matrix reduces the number of function evaluations to adapt to a straight ridge **from** $\mathcal{O}(n^2)$ **to** $\mathcal{O}(n)$.[a]

[a] Hansen, Müller and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation, 11(1)*, pp. 1-18

Number of f-evaluations divided by dimension on the cigar function



The overall model complexity is n^2 but important parts of the model can be learned in time of order n

Rank-μ Update

$$\begin{array}{rclrcl} \boldsymbol{x}_i & = & \boldsymbol{m} + \sigma\, \boldsymbol{y}_i, & \boldsymbol{y}_i & \sim & \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\ \boldsymbol{m} & \leftarrow & \boldsymbol{m} + \sigma \boldsymbol{y}_w & \boldsymbol{y}_w & = & \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \end{array}$$

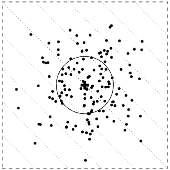
The rank-μ update extends the update rule for **large population sizes** λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation step.
The matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$

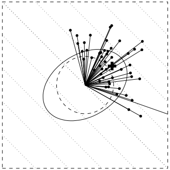
computes a weighted mean of the outer products of the best μ steps and has rank $\min(\mu, n)$ with probability one.
The rank-μ update then reads

$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\, \mathbf{C} + c_{\mathrm{cov}}\, \mathbf{C}_\mu$$

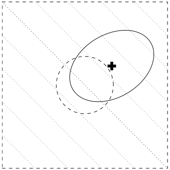
where $c_{\mathrm{cov}} \approx \mu_w / n^2$ and $c_{\mathrm{cov}} \leq 1$.



$$\boldsymbol{x}_i = \boldsymbol{m} + \sigma\, \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$



$$\mathbf{C}_\mu = \tfrac{1}{\mu} \sum \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$
$$\mathbf{C} \leftarrow (1-1) \times \mathbf{C} + 1 \times \mathbf{C}_\mu$$



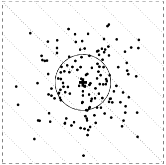
$$\boldsymbol{m}_{\text{new}} \leftarrow \boldsymbol{m} + \tfrac{1}{\mu} \sum \boldsymbol{y}_{i:\lambda}$$

sampling of $\lambda = 150$ solutions where $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$

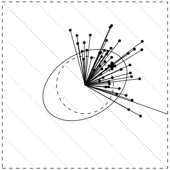
calculating $\mathbf{C}$ where $\mu = 50$, $w_1 = \cdots = w_\mu = \frac{1}{\mu}$, and $c_{\mathrm{cov}} = 1$

new distribution

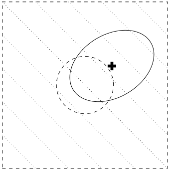
Rank-μ CMA versus Estimation of Multivariate Normal Algorithm EMNA$_{\text{global}}$[6]



$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \mathbf{C})$

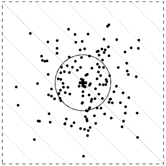


$\mathbf{C} \leftarrow \frac{1}{\mu} \sum (x_{i:\lambda} - \boldsymbol{m}_{\text{old}})(x_{i:\lambda} - \boldsymbol{m}_{\text{old}})^{\mathrm{T}}$

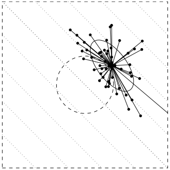


$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu} \sum \boldsymbol{y}_{i:\lambda}$

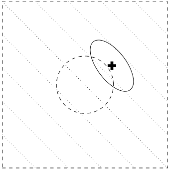
rank-μ CMA conducts a PCA **of steps**



$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \mathbf{C})$



$\mathbf{C} \leftarrow \frac{1}{\mu} \sum (x_{i:\lambda} - \boldsymbol{m}_{\text{new}})(x_{i:\lambda} - \boldsymbol{m}_{\text{new}})^{\mathrm{T}}$



$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu} \sum \boldsymbol{y}_{i:\lambda}$

EMNA$_{\text{global}}$ conducts a PCA **of points**

sampling of $\lambda = 150$ solutions (dots)

calculating $\mathbf{C}$ from $\mu = 50$ solutions

new distribution

The CMA-update yields a larger variance in particular in gradient direction, because $\boldsymbol{m}_{\text{new}}$ is the minimizer for the variances when calculating $\mathbf{C}$

[6] Hansen, N. (2006). The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larranga, I. Inza and E. Bengoetxea (Eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75-102

The **rank-μ update**

- increases the possible learning rate in large populations
  roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary **generations** roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ [7]
  given $\mu_w \propto \lambda \propto n$

Therefore the rank-μ update is the primary mechanism whenever a large population size is used

say $\lambda \geq 3\,n + 10$

The **rank-one update**

- uses the evolution path and reduces the number of necessary **function evaluations** to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank-μ update can be combined

...all equations

[7] Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation, 11(1)*, pp. 1-18

Summary of Equations

The Covariance Matrix Adaptation Evolution Strategy

Input: $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ
Initialize: $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathbf{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
Set: $c_\mathbf{c} \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} \approx 0.3\,\lambda$
While not terminate

$\boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})\,,$ for $i = 1, \ldots, \lambda$	sampling
$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \sigma\boldsymbol{y}_w$ where $\boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i\,\boldsymbol{y}_{i:\lambda}$	update mean
$\boldsymbol{p}_\mathbf{c} \leftarrow (1 - c_\mathbf{c})\,\boldsymbol{p}_\mathbf{c} + \mathbb{1}_{\{\|\boldsymbol{p}_\sigma\| < 1.5\sqrt{n}\}}\sqrt{1 - (1 - c_\mathbf{c})^2}\sqrt{\mu_w}\,\boldsymbol{y}_w$	cumulation for $\mathbf{C}$
$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2}\sqrt{\mu_w}\,\mathbf{C}^{-\frac{1}{2}}\boldsymbol{y}_w$	cumulation for σ
$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\,\mathbf{C} + c_1\,\boldsymbol{p}_\mathbf{c}\boldsymbol{p}_\mathbf{c}^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i\,\boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$	update $\mathbf{C}$
$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$	update of σ

Not covered on this slide: termination, restarts, useful output, boundaries and encoding

Source Code Snippet

W CMA-ES - Wikipedia, t...

http://en.wikipedia.org/wiki/CMA-ES

```
counteval = 0;  % the next 40 lines contain the 20 lines of interesting code
while counteval < stopeval

    % Generate and evaluate lambda offspring
    for k=1:lambda,
        arx(:,k) = xmean + sigma * B * (D .* randn(N,1)); % m + sig * Normal(0,C)
        arfitness(k) = feval(strfitnessfct, arx(:,k)); % objective function call
        counteval = counteval+1;
    end

    % Sort by fitness and compute weighted mean into xmean
    [arfitness, arindex] = sort(arfitness); % minimization
    xold = xmean;
    xmean = arx(:,arindex(1:mu))*weights;   % recombination, new mean value

    % Cumulation: Update evolution paths
    ps = (1-cs)*ps ...
          + sqrt(cs*(2-cs)*mueff) * invsqrtC * (xmean-xold) / sigma;
    hsig = norm(ps)/sqrt(1-(1-cs)^(2*counteval/lambda))/chiN < 1.4 + 2/(N+1);
    pc = (1-cc)*pc ...
          + hsig * sqrt(cc*(2-cc)*mueff) * (xmean-xold) / sigma;

    % Adapt covariance matrix C
    artmp = (1/sigma) * (arx(:,arindex(1:mu))-repmat(xold,1,mu));
    C = (1-c1-cmu) * C ...                   % regard old matrix
         + c1 * (pc*pc' ...                  % plus rank one update
                 + (1-hsig) * cc*(2-cc) * C) ... % minor correction if hsig==0
         + cmu * artmp * diag(weights) * artmp'; % plus rank mu update

    % Adapt step size sigma
    sigma = sigma * exp((cs/damps)*(norm(ps)/chiN - 1));

    % Decomposition of C into B*diag(D.^2)*B' (diagonalization)
    if counteval - eigeneval > lambda/(c1+cmu)/N/10  % to achieve O(N^2)
        eigeneval = counteval;
        C = triu(C) + triu(C,1)'; % enforce symmetry
        [B,D] = eig(C);           % eigen decomposition, B==normalized eigenvectors
        D = sqrt(diag(D));        % D is a vector of standard deviations now
        invsqrtC = B * diag(D.^-1) * B';
    end
```

Evolution Strategies in a Nutshell

1. **Sampling** from a multi-variate normal distribution

   with maximum entropy

2. **Rank-based selection**: same performance on $g(f(\boldsymbol{x}))$ for any g

   $g : \mathbb{R} \to \mathbb{R}$ stricly monotonic (order preserving)

3. **Step-size control**: converge log-linearly on the sphere function and many others possibly with linear scaling in dimension n

4. **Covariance matrix adaptation**: reduce any convex quadratic, g-transformed function

$$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x})$$

   to the sphere function

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

   without use of derivatives

   lines of equal density align with lines of equal fitness $\mathbf{C} \propto \boldsymbol{H}^{-1}$

. . . Theory

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $p(.|\boldsymbol{\theta})$ we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, specifically

- Consider the **natural gradient** of the expected weighted fitness

$$\begin{aligned}\widetilde{\nabla}\,\mathrm{E}(w_g(f(\boldsymbol{x}))|\boldsymbol{\theta}) &= F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathrm{E}(w_g(f(\boldsymbol{x}))|\boldsymbol{\theta}) \\ &= \mathrm{E}(w_g(f(\boldsymbol{x}))F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}|\boldsymbol{\theta}))\end{aligned}$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E\,\frac{\partial^2 \log p(x|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i\partial\boldsymbol{\theta}_j}\right)\right)_{ij}$ of the density p.
  The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

$$\tilde{\nabla}\,\widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i\, F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

CMA-ES = Natural Evolution Strategy + Cumulation

Natural gradient descend using the MC approximation and the normal distribution

- Rewriting the update of the distribution mean

$$\boldsymbol{m}_{\text{new}} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \underbrace{\sum_{i=1}^{\mu} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})}_{\text{natural gradient for mean } \frac{\tilde{\partial}}{\tilde{\partial}\boldsymbol{m}} \widehat{\mathrm{E}}(w_g(f(\boldsymbol{x}))|\boldsymbol{m}, \mathbf{C})}$$

- Rewriting the update of the covariance matrix[8]

$$\mathbf{C}_{\text{new}} \leftarrow \mathbf{C} + c_1 (\overbrace{\boldsymbol{p}_\mathbf{c} \boldsymbol{p}_\mathbf{c}^{\mathrm{T}}}^{\text{rank one}} - \mathbf{C})$$

$$+ \underbrace{\frac{c_\mu}{\sigma^2} \sum_{i=1}^{\mu} w_i \left(\overbrace{(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}}}^{\text{rank-}\mu} - \sigma^2 \mathbf{C} \right)}_{\text{natural gradient for covariance matrix } \frac{\tilde{\partial}}{\tilde{\partial}\mathbf{C}} \widehat{\mathrm{E}}(w_g(f(\boldsymbol{x}))|\boldsymbol{m}, \mathbf{C})}$$

[8] Akimoto et.al. (2010): Bidirectional Relation between CMA Evolution Strategies and Natural Evolution

Maximum Likelihood Update

The new distribution mean $\boldsymbol{m}$ maximizes the log-likelihood

$$\boldsymbol{m}_{\text{new}} = \arg\max_{\boldsymbol{m}} \sum_{i=1}^{\mu} w_i \log p_{\mathcal{N}}(\boldsymbol{x}_{i:\lambda} | \boldsymbol{m})$$

independently of the given covariance matrix

The rank-μ update matrix $\mathbf{C}_\mu$ maximizes the log-likelihood

$$\mathbf{C}_\mu = \arg\max_{\mathbf{C}} \sum_{i=1}^{\mu} w_i \log p_{\mathcal{N}} \left(\frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}_{\text{old}}}{\sigma} \middle| \boldsymbol{m}_{\text{old}}, \mathbf{C} \right)$$

$\log p_{\mathcal{N}}(\boldsymbol{x}|\boldsymbol{m}, \mathbf{C}) = -\frac{1}{2} \log\det(2\pi\mathbf{C}) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m})$
$p_{\mathcal{N}}$ is the density of the multi-variate normal distribution

Variable Metric

On the function class

$$f(\boldsymbol{x}) = g\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^*)^{\mathrm{T}}\right)$$

the covariance matrix approximates the inverse Hessian up to a constant factor, that is:

$$\mathbf{C} \propto \boldsymbol{H}^{-1} \quad \text{(approximately)}$$

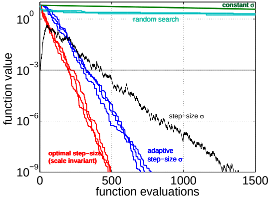
In effect, ellipsoidal level-sets are transformed into spherical level-sets.

$g : \mathbb{R} \to \mathbb{R}$ is strictly increasing

On Convergence

Evolution Strategies converge with probability one on, e.g., $g\left(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}\right)$ like

$$\|\boldsymbol{m}_k - \boldsymbol{x}^*\| \propto e^{-ck}, \qquad c \leq \frac{0.25}{n}$$



Monte Carlo pure random search converges like

$$\|\boldsymbol{m}_k - \boldsymbol{x}^*\| \propto k^{-c} = e^{-c\log k}, \qquad c = \frac{1}{n}$$

Experimentum Crucis (0)

What did we want to achieve?

- reduce any convex-quadratic function

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$$

e.g. $f(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} x_i^2$

to the sphere model

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

without use of derivatives

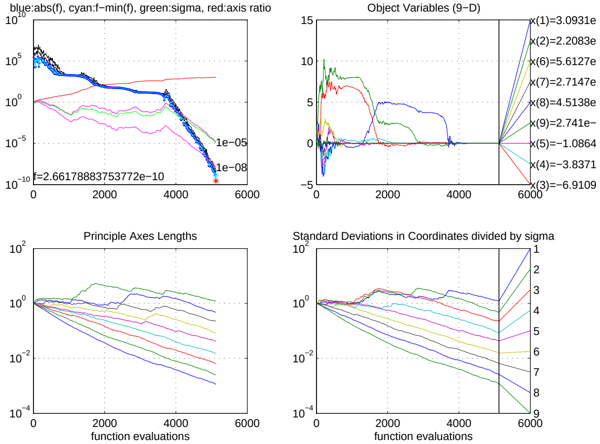
- lines of equal density align with lines of equal fitness

$$\mathbf{C} \propto \boldsymbol{H}^{-1}$$

in a stochastic sense

Experimentum Crucis (1)

f convex quadratic, separable

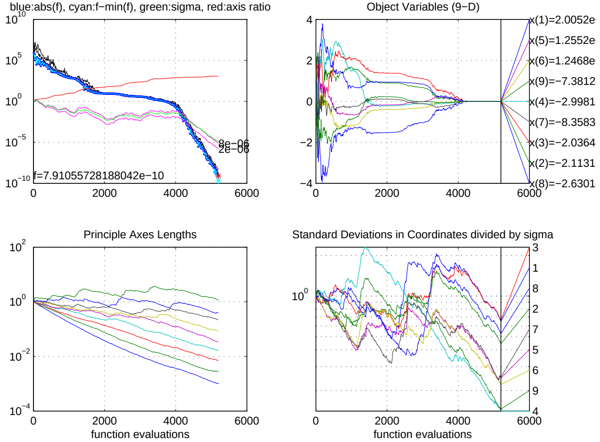


$$f(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{\alpha \frac{i-1}{n-1}} x_i^2, \alpha = 6$$

. . . non-separable

Experimentum Crucis (2)

f convex quadratic, as before but non-separable (rotated)



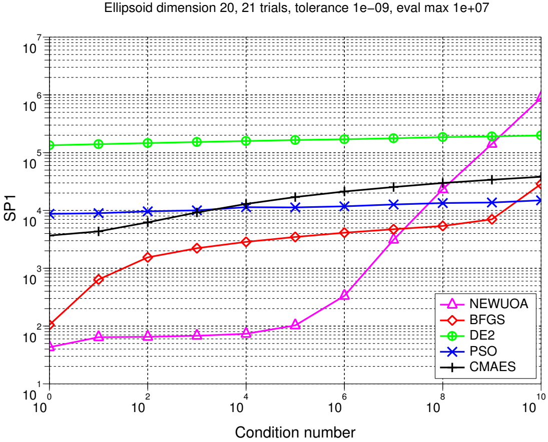
$\mathbf{C} \propto \boldsymbol{H}^{-1}$ **for all** $g, \mathbf{H}$

$f(\boldsymbol{x}) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right)$, $g : \mathbb{R} \to \mathbb{R}$ stricly increasing

...internal parameters

Comparison to BFGS, NEWUOA, PSO and DE

f convex quadratic, separable with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with

$\boldsymbol{H}$ diagonal

g identity (for BFGS and NEWUOA)

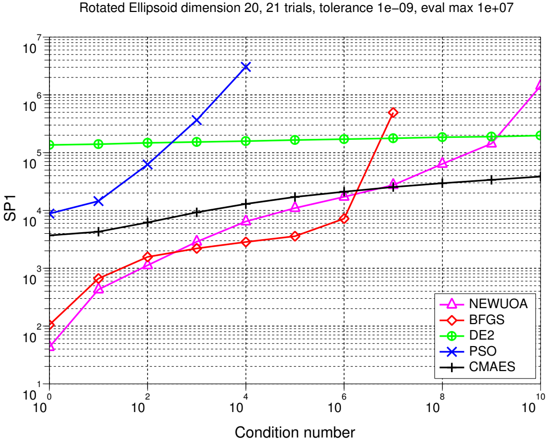
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[9] to reach the target function value of $g^{-1}(10^{-9})$

[9] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

Comparison to BFGS, NEWUOA, PSO and DE

f convex quadratic, non-separable (rotated) with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with

$\boldsymbol{H}$ full

g identity (for BFGS and NEWUOA)

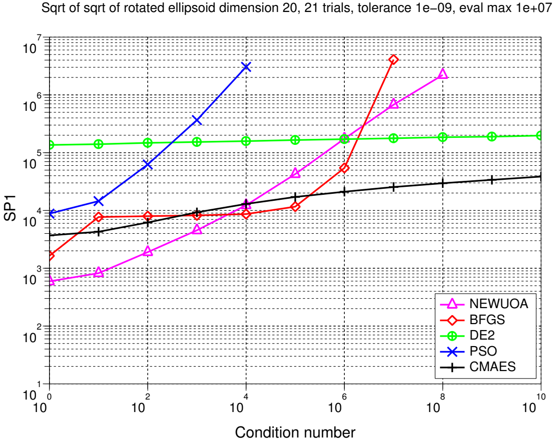
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[10] to reach the target function value of $g^{-1}(10^{-9})$

[10] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

Comparison to BFGS, NEWUOA, PSO and DE

f non-convex, non-separable (rotated) with varying condition number α



BFGS (Broyden et al 1970)
NEWUAO (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with

$\boldsymbol{H}$ full

$g : x \mapsto x^{1/4}$ (for BFGS and NEWUOA)

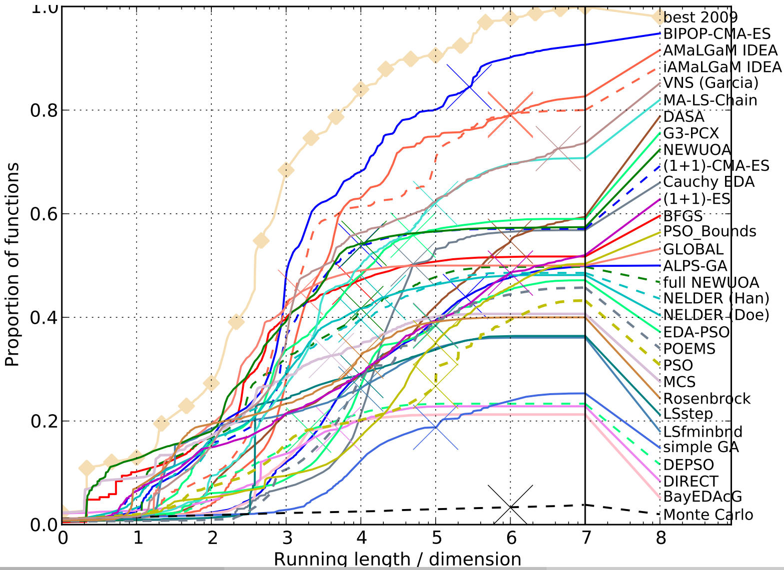
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[11] to reach the target function value of $g^{-1}(10^{-9})$

[11] Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

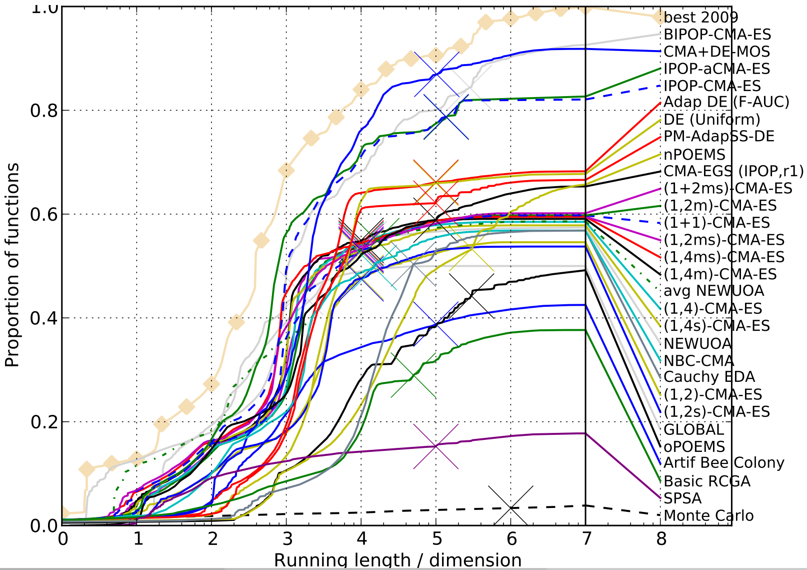
Comparison during BBOB at GECCO 2009

24 functions and 31 algorithms in 20-D



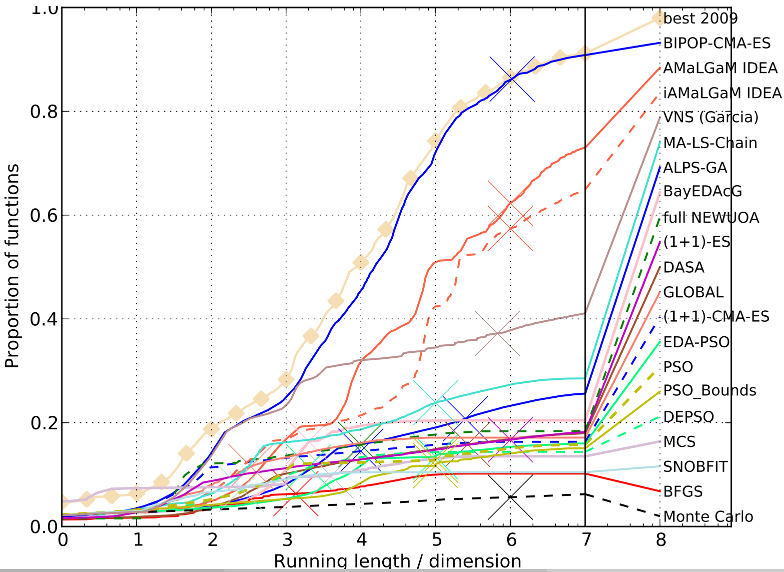
Comparison during BBOB at GECCO 2010

24 functions and 20+ algorithms in 20-D



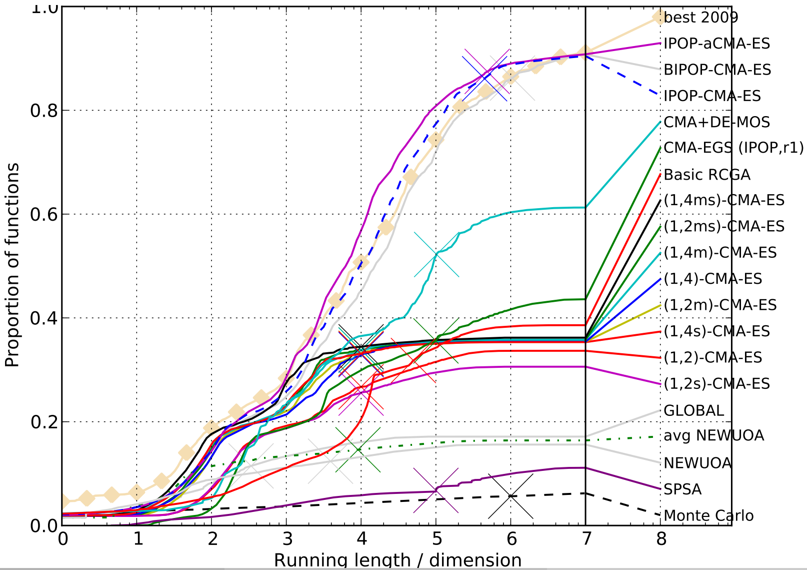
Comparison during BBOB at GECCO 2009

30 **noisy** functions and 20 algorithms in 20-D



Comparison during BBOB at GECCO 2010

30 **noisy** functions and 10+ algorithms in 20-D



The Continuous Search Problem

Difficulties of a non-linear optimization problem are

- dimensionality and non-separabitity
  demands to exploit problem structure, e.g. neighborhood
- ill-conditioning
  demands to acquire a second order model
- ruggedness
  demands a non-local (stochastic? population based?) approach

Main Features of (CMA) Evolution Strategies

1. Multivariate normal distribution to generate new search points
   follows the maximum entropy principle
2. Rank-based selection
   implies invariance, same performance on $g(f(\boldsymbol{x}))$ for any increasing g
   more invariance properties are featured
3. Step-size control facilitates fast (log-linear) convergence and possibly linear scaling with the dimension
   based on an **evolution path** (a non-local trajectory)
4. *Covariance matrix adaptation (CMA)* **increases the likelihood of previously successful steps** and can improve performance by orders of magnitude
   the update follows the natural gradient
   $\mathbf{C} \propto \boldsymbol{H}^{-1} \iff$ adapts a variable metric
   $\iff$ new (rotated) problem representation
   $\implies f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x})$ reduces to $g(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})$

Limitations

of CMA Evolution Strategies

- **internal CPU-time**: $10^{-8}n^2$ seconds per function evaluation on a 2GHz PC, tweaks are available
  1 000 000 f-evaluations in 100-D take 100 seconds *internal* CPU-time
- better methods are presumably available in case of
  - partly separable problems
  - specific problems, for example with cheap gradients
    specific methods
  - small dimension ($n \ll 10$)
    for example Nelder-Mead
  - small running times (number of f-evaluations $\ll 100n$)
    model-based methods

Source code for CMA-ES in C, Java, Matlab, Octave, Scilab, Python is available at
`http://www.lri.fr/~hansen/cmaes_inmatlab.html`