

Evolution Strategies and Covariance Matrix Adaptation

Cours Contrôle Avancé - Ecole Centrale Paris

Anne Auger
January 2014

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

Slides from A. Auger, N. Hansen GECCO 2013 Tutorial on ES and CMA-ES

Content

- 1 Problem Statement
 - Black Box Optimization and Its Difficulties
 - Non-Separable Problems
 - Ill-Conditioned Problems
- 2 Evolution Strategies
 - A Search Template
 - The Normal Distribution
 - Invariance
- 3 Step-Size Control
 - Why Step-Size Control
 - One-Fifth Success Rule
 - Path Length Control (CSA)
- 4 Covariance Matrix Adaptation
 - Covariance Matrix Rank-One Update
 - Cumulation—the Evolution Path
 - Covariance Matrix Rank- μ Update
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks

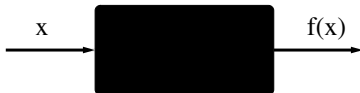
Problem Statement

Continuous Domain Search/Optimization

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

$$f : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}, \quad \mathbf{x} \mapsto f(\mathbf{x})$$

- **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

- solution x with **small function value** $f(x)$ with **least search cost**
 ... or to a robust solution x
 there are two conflicting objectives

● Typical Examples

- shape optimization (e.g. using CFD)
- model calibration
- parameter calibration

curve fitting, airfoils
 biological, physical
 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

Problem Statement

Continuous Domain Search/Optimization

- Goal

- fast convergence to the global optimum

- solution x with **small function value** $f(x)$ with **least search cost**
 ... or to a robust solution x
 there are two conflicting objectives

- Typical Examples

- shape optimization (e.g. using CFD)
- model calibration
- parameter calibration

curve fitting, airfoils
 biological, physical
 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

Problem Statement

Continuous Domain Search/Optimization

- Goal

- fast convergence to the global optimum

- solution x with **small function value** $f(x)$ with **least search cost**
 ... or to a robust solution x
 there are two conflicting objectives

- Typical Examples

- shape optimization (e.g. using CFD)
- model calibration
- parameter calibration

curve fitting, airfoils
 biological, physical
 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 $f : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ 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, plateaus
- ill-conditioned
- noisy
- ...

Goal : cope with any of these function properties

they are related to real-world problems

Objective Function Properties

We assume $f : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ 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, plateaus
- ill-conditioned
- noisy
- ...

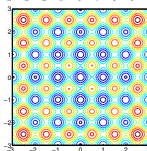
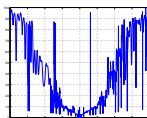
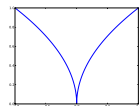
Goal : cope with any of these function properties

they are related to real-world problems

What Makes a Function Difficult to Solve?

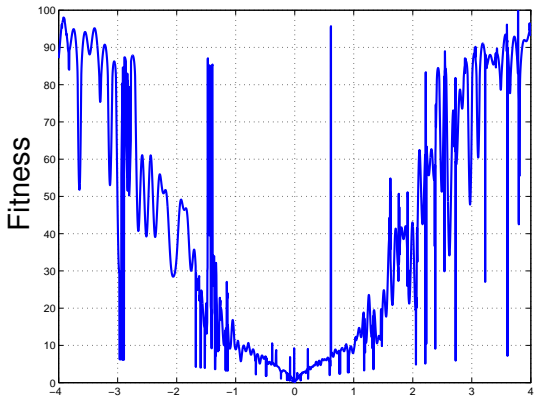
Why stochastic search?

- 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



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)

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

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)

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

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)

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

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)

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

Separable Problems

Definition (Separable Problem)

A function f is separable if

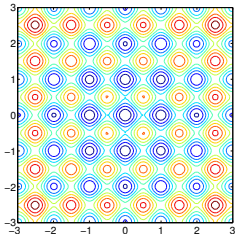
$$\arg \min_{(x_1, \dots, x_n)} f(x_1, \dots, x_n) = \left(\arg \min_{x_1} f(x_1, \dots), \dots, \arg \min_{x_n} f(\dots, x_n) \right)$$

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

Example: Additively decomposable functions

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

Rastrigin function



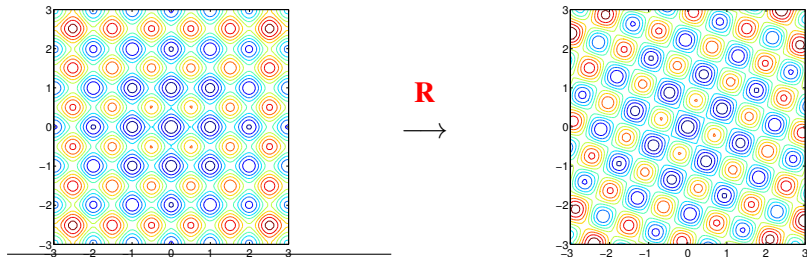
Non-Separable Problems

Building a non-separable problem from a separable one ^(1,2)

Rotating the coordinate system

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

\mathbf{R} rotation matrix



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

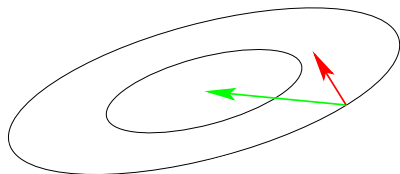
III-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \mathbf{H}(\mathbf{x} - \mathbf{x}^*) = \frac{1}{2} \sum_i h_{i,i} x_i^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} x_i x_j$$

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



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

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

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

What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, 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 population-based method, stochastic, non-elitistic recombination operator serves as repair mechanism restarts

... metaphors

What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, 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 population-based method, stochastic, non-elitistic recombination operator serves as repair mechanism restarts

... metaphors

What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, 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 population-based method, stochastic, non-elitistic recombination operator serves as repair mechanism restarts

... metaphors

Metaphors

Evolutionary Computation

Optimization/Nonlinear Programming

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

- 1 Problem Statement
 - Black Box Optimization and Its Difficulties
 - Non-Separable Problems
 - Ill-Conditioned Problems
- 2 Evolution Strategies
 - A Search Template
 - The Normal Distribution
 - Invariance
- 3 Step-Size Control
 - Why Step-Size Control
 - One-Fifth Success Rule
 - Path Length Control (CSA)
- 4 Covariance Matrix Adaptation
 - Covariance Matrix Rank-One Update
 - Cumulation—the Evolution Path
 - Covariance Matrix Rank- μ Update
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) Estimation of Distribution Algorithms

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) Estimation of Distribution Algorithms

Stochastic Search

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

Initialize distribution parameters θ , set population size $\lambda \in \mathbb{N}$

While not terminate

- 1 Sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
- 2 Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
- 3 Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

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 (incremental) *Estimation of Distribution Algorithms* 

The CMA-ES

Input: $\mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ

Initialize: $\mathbf{C} = \mathbf{I}$, and $\mathbf{p}_c = \mathbf{0}$, $\mathbf{p}_\sigma = \mathbf{0}$,

Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\dots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \approx 0.3 \lambda$

While not terminate

$\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i$, $\mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$, for $i = 1, \dots, \lambda$ sampling

$\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda} = \mathbf{m} + \sigma \mathbf{y}_w$ where $\mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$ update mean

$\mathbf{p}_c \leftarrow (1 - c_c) \mathbf{p}_c + \mathbb{1}_{\{\|\mathbf{p}_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w$ cumulation for \mathbf{C}

$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} \mathbf{C}^{-\frac{1}{2}} \mathbf{y}_w$ cumulation for σ

$\mathbf{C} \leftarrow (1 - c_1 - c_\mu) \mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T$ update \mathbf{C}

$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$ update of σ

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

Evolution Strategies

New search points are sampled normally distributed

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

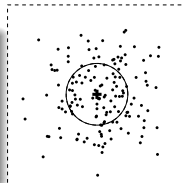
as perturbations of \mathbf{m} , where $\mathbf{x}_i, \mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ 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 \mathbf{m} , \mathbf{C} , and σ .



Evolution Strategies

New search points are sampled normally distributed

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

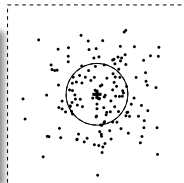
as perturbations of \mathbf{m} , where $\mathbf{x}_i, \mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ 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 \mathbf{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 that the sum of normal variates is again normal:

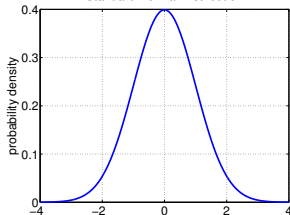
$$\mathcal{N}(\mathbf{x}, \mathbf{A}) + \mathcal{N}(\mathbf{y}, \mathbf{B}) \sim \mathcal{N}(\mathbf{x} + \mathbf{y}, \mathbf{A} + \mathbf{B})$$

helpful in **design and analysis** of algorithms
related to the *central limit theorem*

- 3 most convenient way to generate **isotropic** search points
the isotropic distribution does **not favor any direction**, rotational invariant
- 4 maximum entropy distribution with finite variance
the least possible assumptions on f in the distribution shape

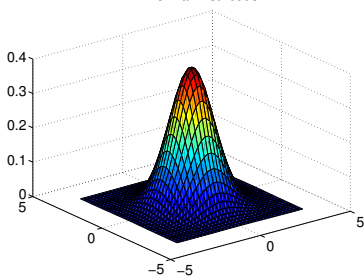
Normal Distribution

Standard Normal Distribution

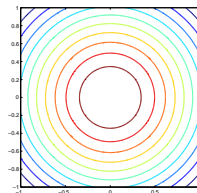


probability density of the 1-D standard normal distribution

2-D Normal Distribution



probability density of a 2-D normal distribution

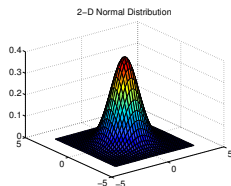


The Multi-Variate (n -Dimensional) Normal Distribution

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

The mean value \mathbf{m}

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

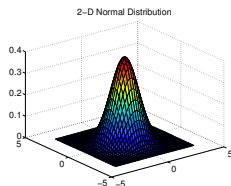


The Multi-Variate (n -Dimensional) Normal Distribution

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

The **mean** value \mathbf{m}

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

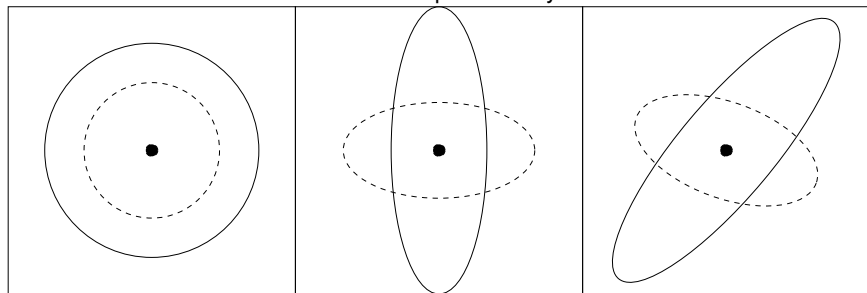


The **covariance matrix** \mathbf{C}

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

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

Lines of Equal Density



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

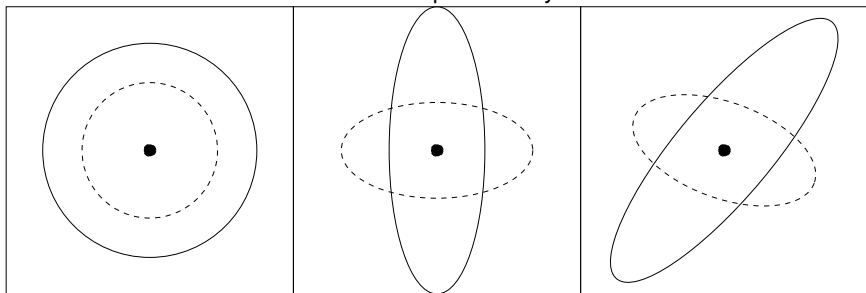
$\mathcal{N}(\mathbf{m}, \mathbf{D}^2) \sim \mathbf{m} + \mathbf{D} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 n degrees of freedom
 components are
 independent, scaled

$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $(n^2 + n)/2$ 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 $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^T)$ holds for all \mathbf{A} .

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

Lines of Equal Density



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

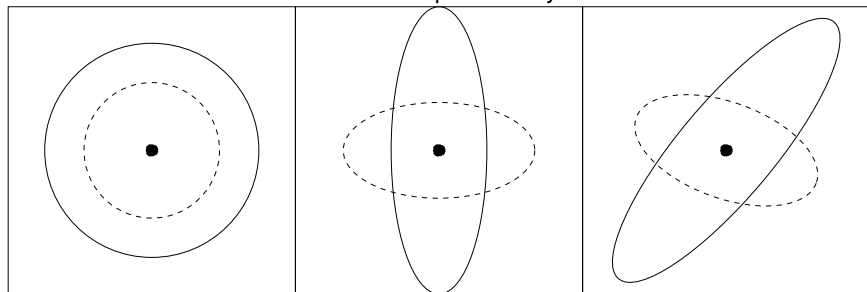
$\mathcal{N}(\mathbf{m}, \mathbf{D}^2) \sim \mathbf{m} + \mathbf{D} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 n degrees of freedom
 components are
 independent, scaled

$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $(n^2 + n)/2$ 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 $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^T)$ holds for all \mathbf{A} .

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

Lines of Equal Density



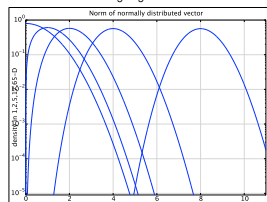
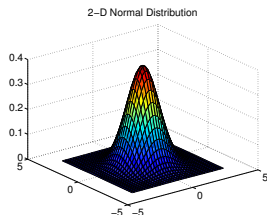
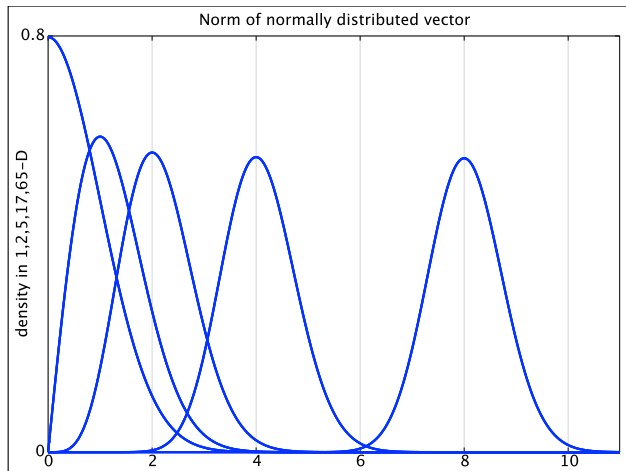
$\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{I}) \sim \mathbf{m} + \sigma \mathcal{N}(\mathbf{0}, \mathbf{I})$
one degree of freedom σ
 components are
 independent standard
 normally distributed

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

$\mathcal{N}(\mathbf{m}, \mathbf{C}) \sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $(n^2 + n)/2$ 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 $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^T)$ holds for all \mathbf{A} .

Effect of Dimensionality



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

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

yet: maximum entropy distribution

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 m

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

If \mathbf{x} better than m select

$$m \leftarrow \mathbf{x}$$

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

Non-elitist selection and intermediate (weighted) recombination

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

Let $\mathbf{x}_{i:\lambda}$ the i -th ranked solution point, such that $f(\mathbf{x}_{1:\lambda}) \leq \dots \leq f(\mathbf{x}_{\lambda:\lambda})$.

The new mean reads

$$\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$$

where

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

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

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

Non-elitist selection and intermediate (weighted) recombination

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

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

$$\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$$

where

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

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

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

Non-elitist selection and intermediate (weighted) recombination

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

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

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

where

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

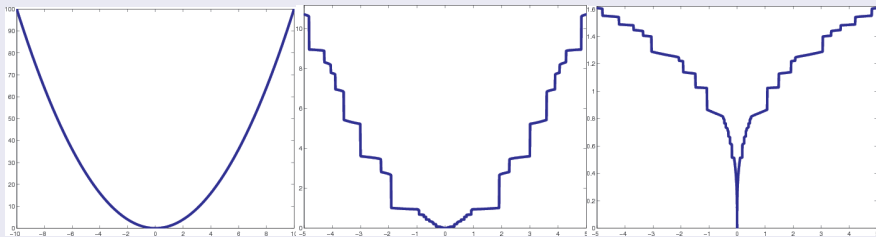
The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

Invariance Under Monotonically Increasing Functions

Rank-based algorithms

Update of all parameters uses only the ranks

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



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

g is strictly monotonically increasing
 g preserves ranks

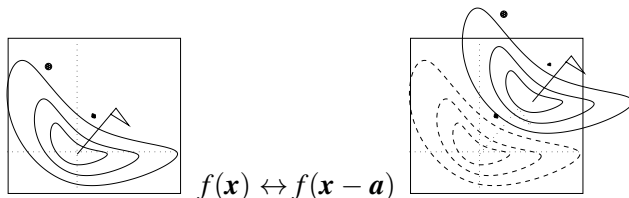
3

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



Identical behavior on f and f_a

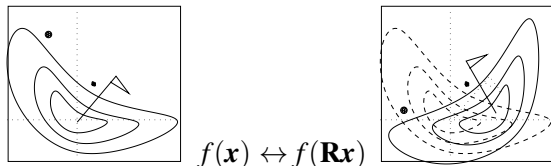
$$f : \mathbf{x} \mapsto f(\mathbf{x}), \quad \mathbf{x}^{(t=0)} = \mathbf{x}_0$$

$$f_a : \mathbf{x} \mapsto f(\mathbf{x} - \mathbf{a}), \quad \mathbf{x}^{(t=0)} = \mathbf{x}_0 + \mathbf{a}$$

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 $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
 e.g. true for simple evolution strategies
 recombination operators might jeopardize rotational invariance



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

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

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

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

- 1 Problem Statement
 - Black Box Optimization and Its Difficulties
 - Non-Separable Problems
 - Ill-Conditioned Problems
- 2 Evolution Strategies
 - A Search Template
 - The Normal Distribution
 - Invariance
- 3 Step-Size Control**
 - Why Step-Size Control
 - One-Fifth Success Rule
 - Path Length Control (CSA)
- 4 Covariance Matrix Adaptation
 - Covariance Matrix Rank-One Update
 - Cumulation—the Evolution Path
 - Covariance Matrix Rank- μ Update
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks

Evolution Strategies

Recalling

New search points are sampled normally distributed

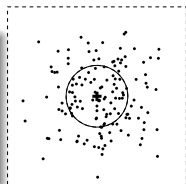
$$\mathbf{x}_i \sim \mathbf{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

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

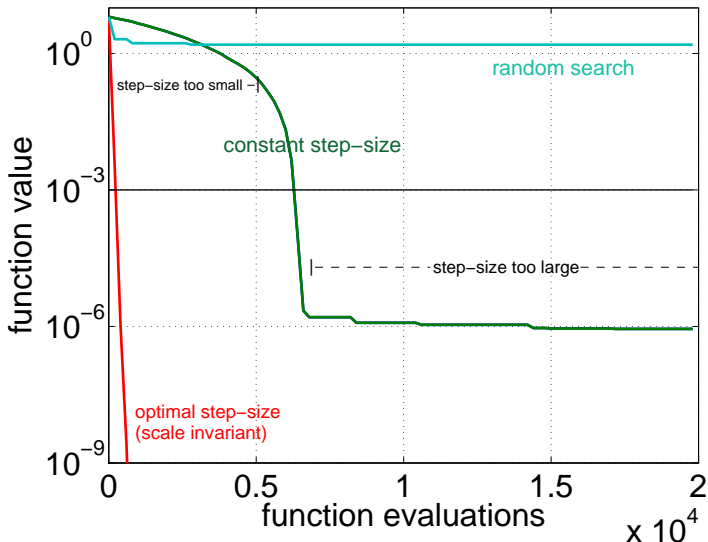
where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ represents the favorite solution and $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$
- 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?



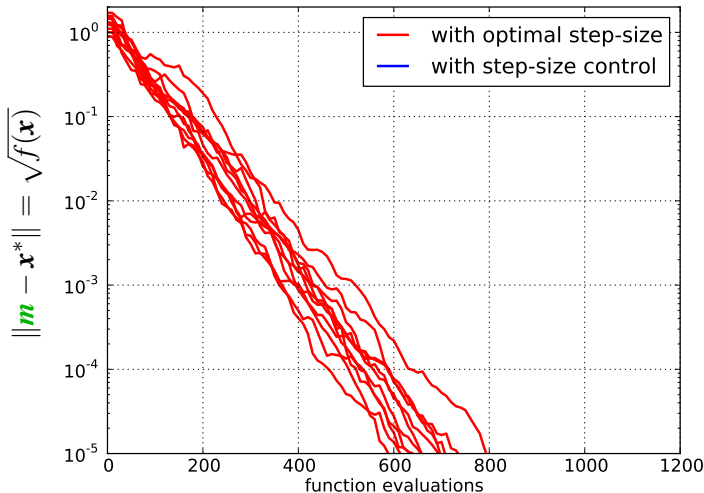
(1+1)-ES
(red & green)

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

in $[-2.2, 0.8]^n$
for $n = 10$

Why Step-Size Control?

(5/5_w, 10)-ES, 11 runs



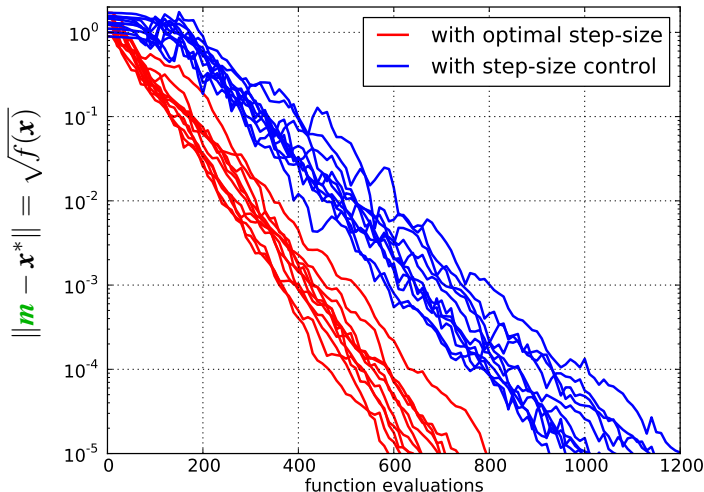
$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$$

for $n = 10$ and
 $\mathbf{x}^0 \in [-0.2, 0.8]^n$

with optimal step-size σ

Why Step-Size Control?

(5/5_w, 10)-ES, 2×11 runs



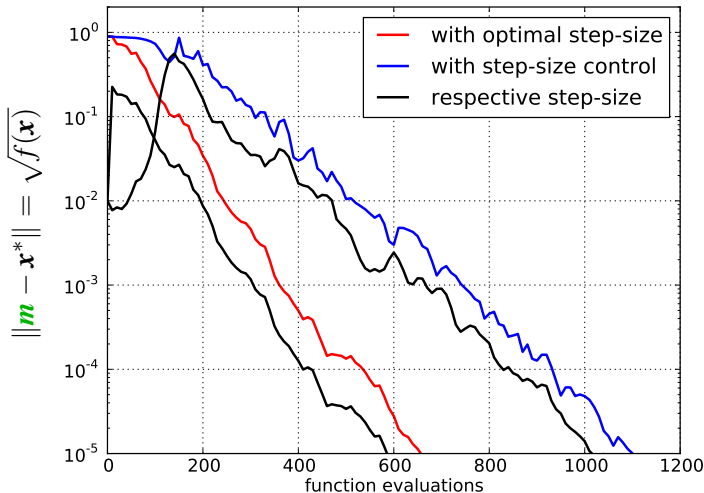
$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$$

for $n = 10$ and
 $\mathbf{x}^0 \in [-0.2, 0.8]^n$

with **optimal** versus **adaptive** step-size σ with too small initial σ

Why Step-Size Control?

(5/5_w, 10)-ES



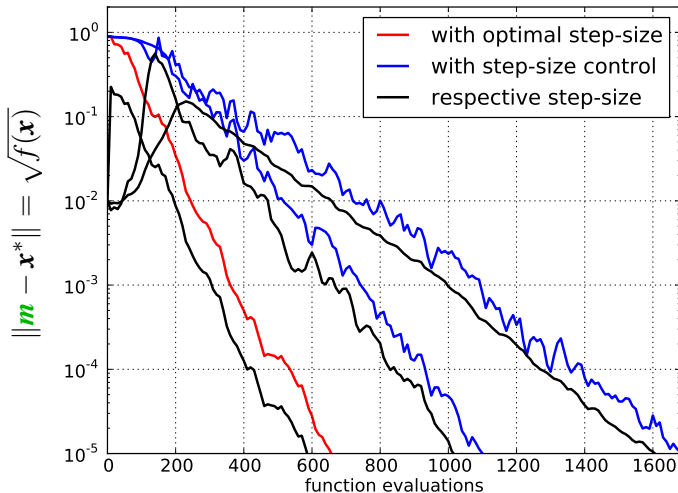
$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$$

for $n = 10$ and
 $\mathbf{x}^0 \in [-0.2, 0.8]^n$

comparing number of f -evals to reach $\|m\| = 10^{-5}$: $\frac{1100-100}{650} \approx 1.5$

Why Step-Size Control?

(5/5_w, 10)-ES

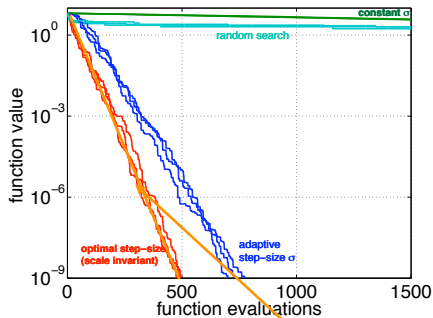


$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$$

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

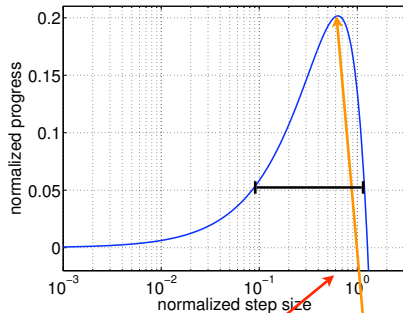
comparing optimal versus default damping parameter $d_\sigma: \frac{1700}{1100} \approx 1.5$

Why Step-Size Control?



$$\sigma \leftarrow \sigma_{\text{opt}}^* \|\text{parent}\|$$

$$\frac{\varphi^*}{n}$$



$$\sigma_{\text{opt}}^*$$

$$\varphi^*$$

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

Methods for Step-Size Control

- **1/5-th success rule^{ab}**, 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

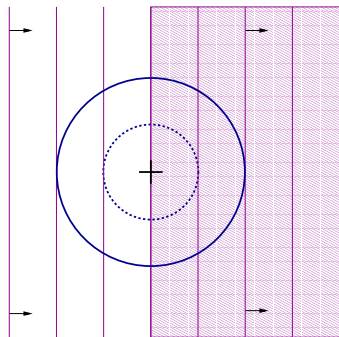
^aRechenberg 1973, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog

^bSchumer and Steiglitz 1968. Adaptive step size random search. *IEEE TAC*

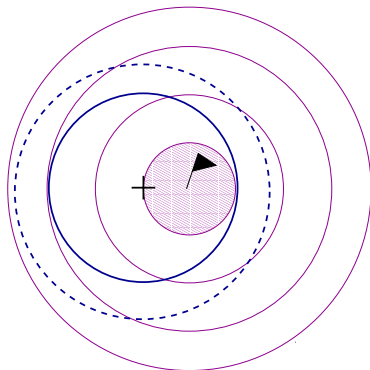
^cSchwefel 1981, *Numerical Optimization of Computer Models*, Wiley

^dHansen & Ostermeier 2001, Completely Derandomized Self-Adaptation in Evolution Strategies, *Evol. Comput.*

One-fifth success rule

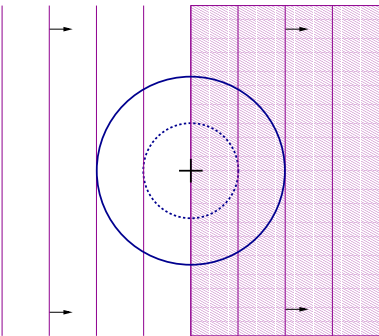


increase σ



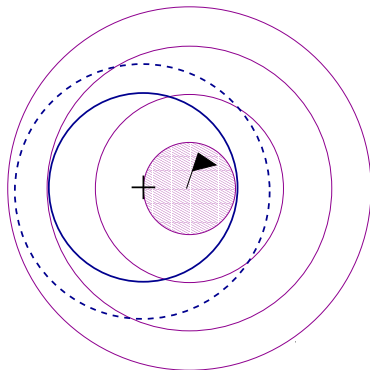
decrease σ

One-fifth success rule



Probability of success (p_s)

$1/2$



Probability of success (p_s)

“too small”

$1/5$

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

$$p_{\text{target}} = 1/5$$

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)

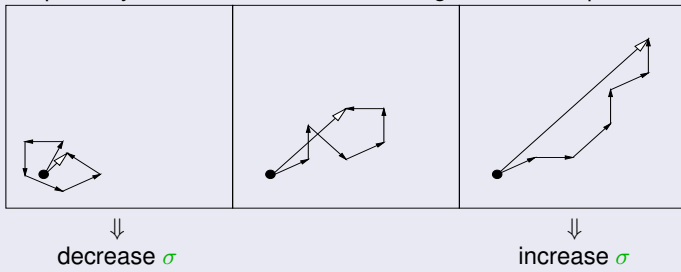
The Concept of Cumulative Step-Size Adaptation

$$x_i = m + \sigma y_i$$

$$m \leftarrow m + \sigma y_w$$

Measure the length of the *evolution path*

the pathway of the mean vector m in the generation sequence



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

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

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

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

Path Length Control (CSA)

The Equations

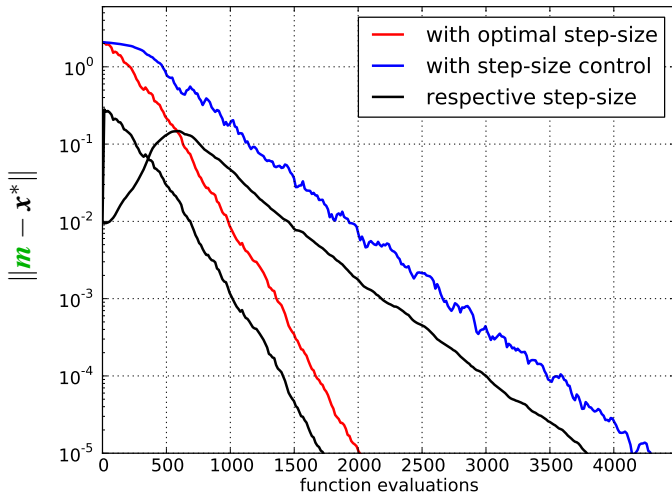
Initialize $\mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\mathbf{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

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

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

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

(5/5, 10)-CSA-ES, default parameters



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

in $[-0.2, 0.8]^n$
for $n = 30$

- 1 Problem Statement
- 2 Evolution Strategies
- 3 Step-Size Control
- 4 Covariance Matrix Adaptation**
 - Covariance Matrix Rank-One Update
 - Cumulation—the Evolution Path
 - Covariance Matrix Rank- μ Update
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks

Evolution Strategies

Recalling

New search points are sampled normally distributed

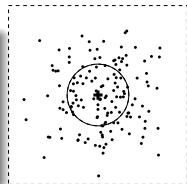
$$\mathbf{x}_i \sim \mathbf{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \quad \text{for } i = 1, \dots, \lambda$$

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

where

- the **mean** vector $\mathbf{m} \in \mathbb{R}^n$ 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

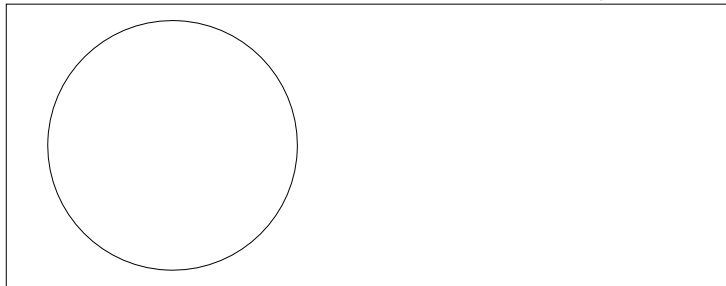
The remaining question is how to update \mathbf{C} .



Covariance Matrix Adaptation

Rank-One Update

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

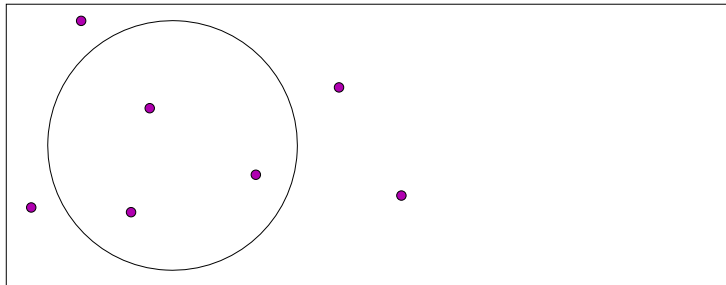


initial distribution, $\mathbf{C} = \mathbf{I}$

Covariance Matrix Adaptation

Rank-One Update

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

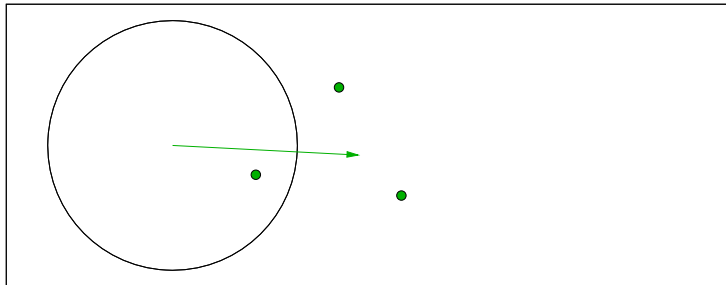


initial distribution, $\mathbf{C} = \mathbf{I}$

Covariance Matrix Adaptation

Rank-One Update

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

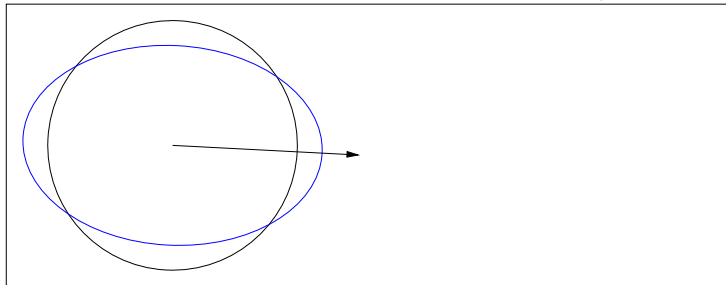


\mathbf{y}_w , movement of the population mean \mathbf{m} (disregarding σ)

Covariance Matrix Adaptation

Rank-One Update

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



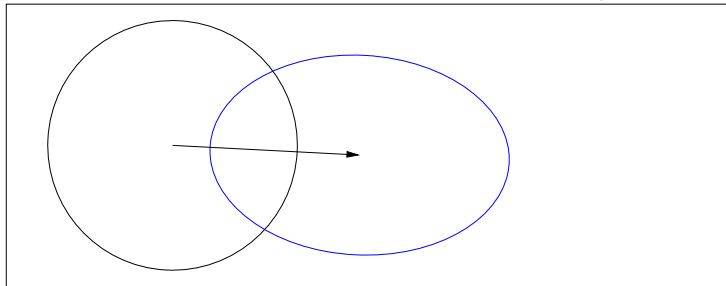
mixture of distribution \mathbf{C} and step \mathbf{y}_w ,

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

Covariance Matrix Adaptation

Rank-One Update

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

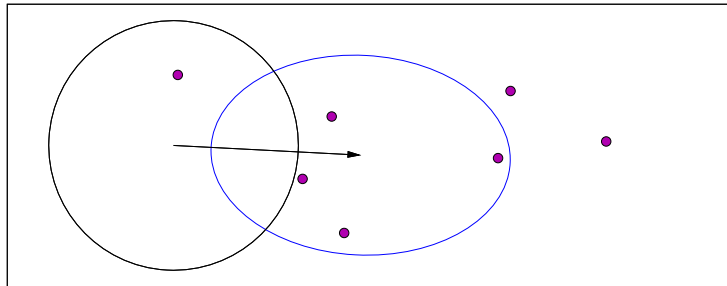


new distribution (disregarding σ)

Covariance Matrix Adaptation

Rank-One Update

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

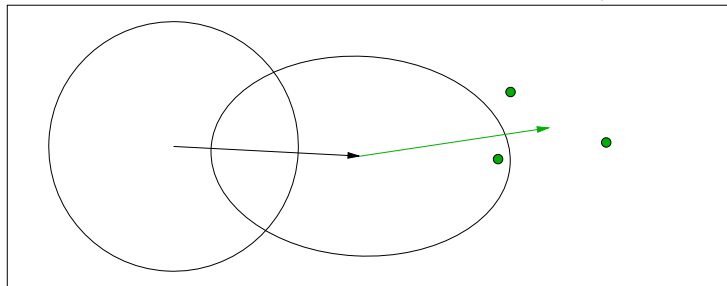


new distribution (disregarding σ)

Covariance Matrix Adaptation

Rank-One Update

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

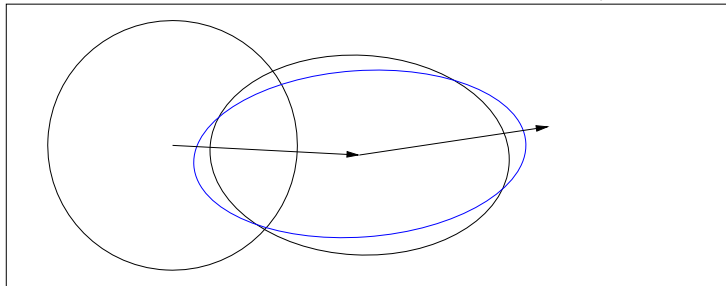


movement of the population mean \mathbf{m}

Covariance Matrix Adaptation

Rank-One Update

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



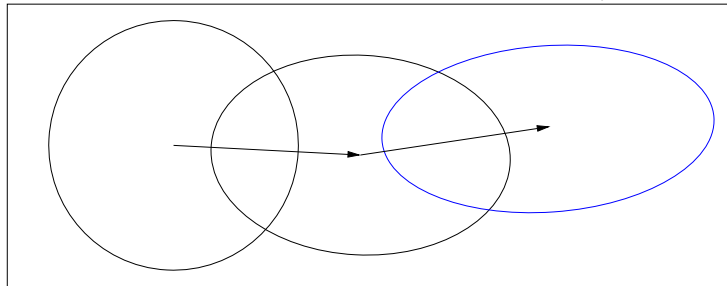
mixture of distribution \mathbf{C} and step \mathbf{y}_w ,

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

Covariance Matrix Adaptation

Rank-One Update

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



new distribution,

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

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

another viewpoint: the adaptation **follows a natural gradient**

approximation of the expected fitness

Covariance Matrix Adaptation

Rank-One Update

Initialize $\mathbf{m} \in \mathbb{R}^n$, and $\mathbf{C} = \mathbf{I}$, set $\sigma = 1$, learning rate $c_{\text{cov}} \approx 2/n^2$

While not terminate

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

$$\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w \quad \text{where } \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$$

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\mu_w \mathbf{y}_w \mathbf{y}_w^T}_{\text{rank-one}} \quad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$$

The rank-one update has been found independently in several domains^{6 7 8 9}

⁶ Kjellström&Taxén 1981. Stochastic Optimization in System Design, IEEE TCS

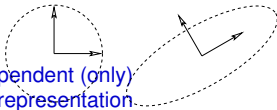
⁷ Hansen&Ostermeier 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, ICEC

⁸ Ljung 1999. System Identification: Theory for the User

⁹ Haario et al 2001. An adaptive Metropolis algorithm, JSTOR

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

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 \mathbf{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
- learns a new **rotated problem representation**
components are independent (only) in the new representation 
- learns a **new** (Mahalanobis) **metric**
variable metric method
- approximates the **inverse Hessian** on quadratic functions
transformation into the sphere function
- for $\mu = 1$: conducts a **natural gradient ascent** on the distribution \mathcal{N}
entirely independent of the given coordinate system

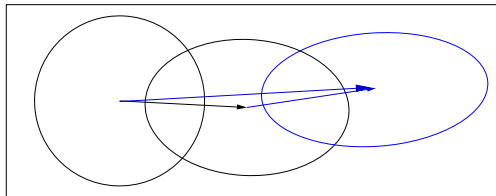
- 1 Problem Statement
- 2 Evolution Strategies
- 3 Step-Size Control
- 4 Covariance Matrix Adaptation**
 - Covariance Matrix Rank-One Update
 - Cumulation—the Evolution Path
 - Covariance Matrix Rank- μ Update
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks

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 m .



An exponentially weighted sum of steps y_w is used

$$p_c \propto \sum_{i=0}^g \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} y_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} p_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \underbrace{y_w}_{\text{input} = \frac{m - m_{\text{old}}}{\sigma}}$$

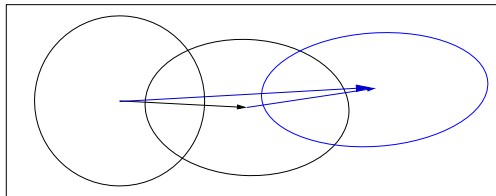
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. **History information** is accumulated in the evolution path.

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 m .



An exponentially weighted sum of steps y_w is used

$$p_c \propto \sum_{i=0}^g \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} y_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} p_c + \underbrace{\sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}}_{\text{normalization factor}} \underbrace{y_w}_{\text{input} = \frac{m - m_{\text{old}}}{\sigma}}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. **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 *moving average*
- *iterate averaging* in stochastic approximation
- *momentum* in the back-propagation algorithm for ANNs
- ...

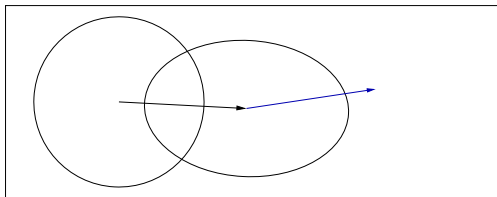
“Cumulation” conducts a *low-pass* filtering, but there is more to it. . .

...why?

Cumulation

Utilizing the Evolution Path

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



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

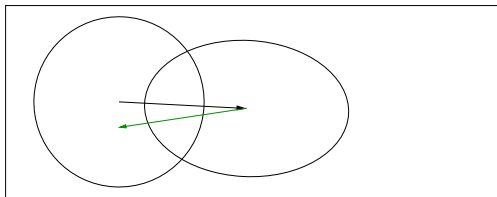
$$\begin{aligned}
 \mathbf{p}_c &\leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \mathbf{y}_w \\
 \mathbf{C} &\leftarrow (1 - c_{\text{cov}}) \mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c \mathbf{p}_c^T}_{\text{rank-one}}
 \end{aligned}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the “backward time horizon”.

Cumulation

Utilizing the Evolution Path

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



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

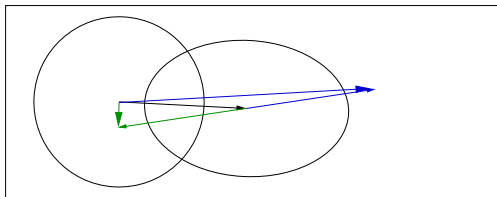
$$\begin{aligned}
 \mathbf{p}_c &\leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \mathbf{y}_w \\
 \mathbf{C} &\leftarrow (1 - c_{\text{cov}}) \mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c \mathbf{p}_c^T}_{\text{rank-one}}
 \end{aligned}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the “backward time horizon”.

Cumulation

Utilizing the Evolution Path

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



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

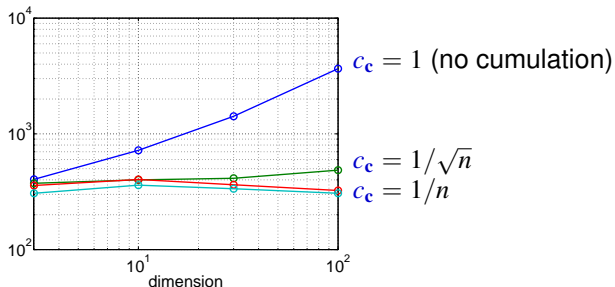
$$\begin{aligned}
 \mathbf{p}_c &\leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \mathbf{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2}}_{\text{normalization factor}} \sqrt{\mu_w} \mathbf{y}_w \\
 \mathbf{C} &\leftarrow (1 - c_{\text{cov}}) \mathbf{C} + c_{\text{cov}} \underbrace{\mathbf{p}_c \mathbf{p}_c^T}_{\text{rank-one}}
 \end{aligned}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\text{cov}} \ll c_c \ll 1$ such that $1/c_c$ 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 about $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$** .^(a)

^aHansen, 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 $f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^n x_i^2$



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

Rank- μ Update

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

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 weighted empirical covariance matrix

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

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

with $\mu = \lambda$ weights can be negative ¹⁰

The rank- μ update then reads

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

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

¹⁰ Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation.

Rank- μ Update

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

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 weighted empirical covariance matrix

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

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

with $\mu = \lambda$ weights can be negative ¹⁰

The rank- μ update then reads

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

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

¹⁰ Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC.

Rank- μ Update

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

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 weighted empirical covariance matrix

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

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

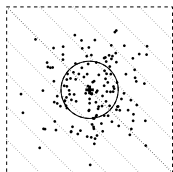
with $\mu = \lambda$ weights can be negative ¹⁰

The rank- μ update then reads

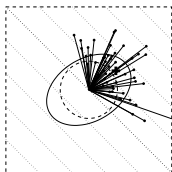
$$\mathbf{C} \leftarrow (1 - c_{\text{cov}}) \mathbf{C} + c_{\text{cov}} \mathbf{C}_\mu$$

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

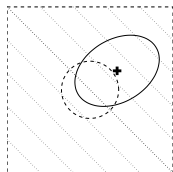
¹⁰ Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation.



$$x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$



$$\begin{aligned} \mathbf{C}_\mu &= \frac{1}{\mu} \sum y_{i:\lambda} y_{i:\lambda}^T \\ \mathbf{C} &\leftarrow \frac{1}{(1-\lambda)} \times \mathbf{C} + \lambda \times \mathbf{C}_\mu \end{aligned}$$

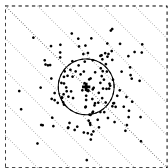


$$m_{\text{new}} \leftarrow m + \frac{1}{\mu} \sum y_{i:\lambda}$$

new distribution

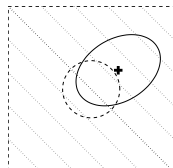
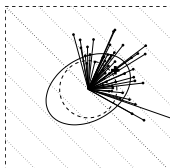
sampling of $\lambda = 150$
solutions where
 $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$

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

Rank- μ CMA versus Estimation of Multivariate Normal Algorithm EMNA_{global}¹¹

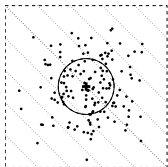
$$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(0, \mathbf{C})$$

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



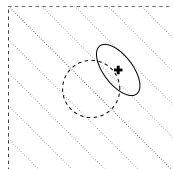
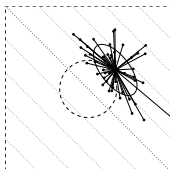
$$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum y_{i:\lambda}$$

rank- μ CMA
conducts a
PCA of
steps



$$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(0, \mathbf{C})$$

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



$$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum y_{i:\lambda}$$

EMNA_{global}
conducts a
PCA of
points

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

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

new distribution

m_{new} is the minimizer for the variances when calculating \mathbf{C}

¹¹ 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 $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

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

say $\lambda \geq 3n + 10$

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

¹²Hansen, 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

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 $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

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

say $\lambda \geq 3n + 10$

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

¹²Hansen, 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

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 $\mathcal{O}(n)$ ⁽¹²⁾
given $\mu_w \propto \lambda \propto n$

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

say $\lambda \geq 3n + 10$

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

¹²Hansen, 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: $\mathbf{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ

Initialize: $\mathbf{C} = \mathbf{I}$, and $\mathbf{p}_c = \mathbf{0}$, $\mathbf{p}_\sigma = \mathbf{0}$,

Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\dots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^\lambda w_i^2} \approx 0.3 \lambda$

While not terminate

$\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i$, $\mathbf{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$, for $i = 1, \dots, \lambda$ sampling

$\mathbf{m} \leftarrow \sum_{i=1}^\mu w_i \mathbf{x}_{i:\lambda} = \mathbf{m} + \sigma \mathbf{y}_w$ where $\mathbf{y}_w = \sum_{i=1}^\mu w_i \mathbf{y}_{i:\lambda}$ update mean

$\mathbf{p}_c \leftarrow (1 - c_c) \mathbf{p}_c + \mathbb{1}_{\{\|\mathbf{p}_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w$ cumulation for \mathbf{C}

$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} \mathbf{C}^{-\frac{1}{2}} \mathbf{y}_w$ cumulation for σ

$\mathbf{C} \leftarrow (1 - c_1 - c_\mu) \mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^\mu w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T$ update \mathbf{C}

$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$ update of σ

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

Source Code Snippet

```

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-cl-cmu) * C ... % regard old matrix
        + cl * (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/(cl+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
end

```


Strategy Internal Parameters

- related to selection and recombination
 - λ , offspring number, new solutions sampled, population size
 - μ , parent number, solutions involved in updates of m , C , and σ
 - $w_{i=1,\dots,\mu}$, recombination weights
- related to C -update
 - c_c , decay rate for the evolution path
 - c_1 , learning rate for rank-one update of C
 - c_μ , learning rate for rank- μ update of C
- related to σ -update
 - c_σ , decay rate of the evolution path
 - d_σ , damping for σ -change

Parameters were identified in carefully chosen experimental set ups. **Parameters do not in the first place depend on the objective function** and are not meant to be in the users choice.

Only(?) the population size λ might be reasonably varied in a wide range, *depending on the objective function*

Useful: restarts with increasing population size (IPOP)

Experimentum Crucis (0)

What did we want to achieve?

- reduce any convex-quadratic function

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

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

to the sphere model

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$$

without use of derivatives

- lines of equal density align with lines of equal fitness

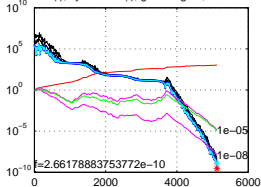
$$\mathbf{C} \propto \mathbf{H}^{-1}$$

in a stochastic sense

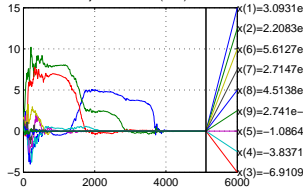
Experimentum Crucis (1)

f convex quadratic, separable

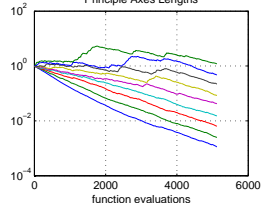
blue:abs(f), cyan:f-min(f), green:sigma, red:axis ratio



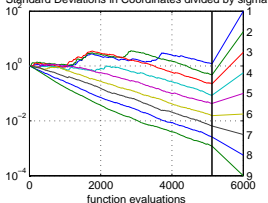
Object Variables (9-D)



Principle Axes Lengths



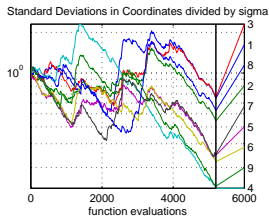
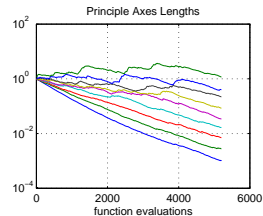
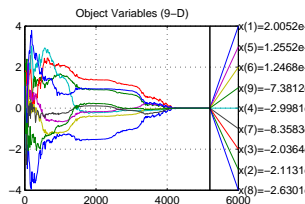
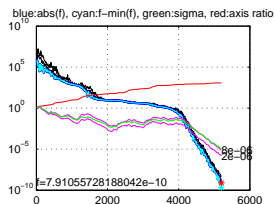
Standard Deviations in Coordinates divided by sigma



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

Experimentum Crucis (2)

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



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

$f(x) = g(x^T H x)$, $g : \mathbb{R} \rightarrow \mathbb{R}$ strictly increasing

- 1 Problem Statement
- 2 Evolution Strategies
- 3 Step-Size Control
- 4 Covariance Matrix Adaptation
- 5 CMA-ES Summary
- 6 Theoretical Foundations**
- 7 Comparing Experiments
- 8 Summary and Final Remarks

Natural Gradient Descent

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

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathbb{E}(f(\mathbf{x})|\theta), \quad \eta > 0$$

∇_{θ} depends on the parameterization of the distribution, therefore

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

$$\begin{aligned} \tilde{\nabla}_{\theta} \mathbb{E}(w \circ P_f(f(\mathbf{x}))|\theta) &= F_{\theta}^{-1} \nabla_{\theta} \mathbb{E}(w \circ P_f(f(\mathbf{x}))|\theta) \\ &= \mathbb{E}(w \circ P_f(f(\mathbf{x})) F_{\theta}^{-1} \nabla_{\theta} \ln p(\mathbf{x}|\theta)) \end{aligned}$$

using the Fisher information matrix $F_{\theta} = \left(\mathbb{E} \left(\frac{\partial^2 \log p(\mathbf{x}|\theta)}{\partial \theta_i \partial \theta_j} \right) \right)_{ij}$ of the density p .

The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

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

Natural Gradient Descent

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

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathbb{E}(f(\mathbf{x})|\theta), \quad \eta > 0$$

∇_{θ} depends on the parameterization of the distribution, therefore

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

$$\begin{aligned} \tilde{\nabla}_{\theta} \mathbb{E}(w \circ P_f(f(\mathbf{x}))|\theta) &= F_{\theta}^{-1} \nabla_{\theta} \mathbb{E}(w \circ P_f(f(\mathbf{x}))|\theta) \\ &= \mathbb{E}(w \circ P_f(f(\mathbf{x})) F_{\theta}^{-1} \nabla_{\theta} \ln p(\mathbf{x}|\theta)) \end{aligned}$$

using the Fisher information matrix $F_{\theta} = \left(\mathbb{E} \left(\frac{\partial^2 \log p(\mathbf{x}|\theta)}{\partial \theta_i \partial \theta_j} \right) \right)_{ij}$ of the density p .

The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

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

Natural Gradient Descent

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

$$\theta \leftarrow \theta - \eta \nabla_{\theta} E(f(\mathbf{x})|\theta), \quad \eta > 0$$

∇_{θ} depends on the parameterization of the distribution, therefore

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

$$\begin{aligned} \tilde{\nabla}_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) &= F_{\theta}^{-1} \nabla_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) \\ &= E(w \circ P_f(f(\mathbf{x})) F_{\theta}^{-1} \nabla_{\theta} \ln p(\mathbf{x}|\theta)) \end{aligned}$$

using the Fisher information matrix $F_{\theta} = \left(\left(E \frac{\partial^2 \log p(\mathbf{x}|\theta)}{\partial \theta_i \partial \theta_j} \right) \right)_{ij}$ of the density p .

The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

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

Natural Gradient Descent

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

$$\theta \leftarrow \theta - \eta \nabla_{\theta} E(f(\mathbf{x})|\theta), \quad \eta > 0$$

∇_{θ} depends on the parameterization of the distribution, therefore

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

$$\begin{aligned} \tilde{\nabla}_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) &= F_{\theta}^{-1} \nabla_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) \\ &= E(w \circ P_f(f(\mathbf{x})) F_{\theta}^{-1} \nabla_{\theta} \ln p(\mathbf{x}|\theta)) \end{aligned}$$

using the Fisher information matrix $F_{\theta} = \left(\left(E \frac{\partial^2 \log p(\mathbf{x}|\theta)}{\partial \theta_i \partial \theta_j} \right) \right)_{ij}$ of the density p .

The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

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

Natural Gradient Descent

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

$$\theta \leftarrow \theta - \eta \nabla_{\theta} E(f(\mathbf{x})|\theta), \quad \eta > 0$$

∇_{θ} depends on the parameterization of the distribution, therefore

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

$$\begin{aligned} \tilde{\nabla}_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) &= F_{\theta}^{-1} \nabla_{\theta} E(w \circ P_f(f(\mathbf{x}))|\theta) \\ &= E(w \circ P_f(f(\mathbf{x})) F_{\theta}^{-1} \nabla_{\theta} \ln p(\mathbf{x}|\theta)) \end{aligned}$$

using the Fisher information matrix $F_{\theta} = \left(\left(E \frac{\partial^2 \log p(\mathbf{x}|\theta)}{\partial \theta_i \partial \theta_j} \right) \right)_{ij}$ of the density p .

The natural gradient is **invariant under re-parameterization** of the distribution.

- A **Monte-Carlo approximation** reads

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

CMA-ES – Cumulation = Natural Evolution Strategy

Natural gradient descend using the MC approximation and the normal distribution

- Rewriting the update of the distribution mean

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

- Rewriting the update of the covariance matrix¹³

$$\begin{aligned} \mathbf{C}_{\text{new}} \leftarrow & \mathbf{C} + c_1 \overbrace{(\mathbf{p}_c \mathbf{p}_c^T)}^{\text{rank one}} - \mathbf{C} \\ & + \frac{c_\mu}{\sigma^2} \sum_{i=1}^{\mu} w_i \underbrace{\left((\mathbf{x}_{i:\lambda} - \mathbf{m}) (\mathbf{x}_{i:\lambda} - \mathbf{m})^T - \sigma^2 \mathbf{C} \right)}_{\text{rank-}\mu} \\ & \text{natural gradient for covariance matrix } \frac{\partial}{\partial \mathbf{C}} \widehat{\mathbb{E}}(w \circ P_f(f(\mathbf{x})) | \mathbf{m}, \mathbf{C}) \end{aligned}$$

13

Maximum Likelihood Update

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

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

independently of the given covariance matrix

$$\log p_{\mathcal{N}}(\mathbf{x} | \mathbf{m}, \mathbf{C}) = -\frac{1}{2} \log \det(2\pi\mathbf{C}) - \frac{1}{2}(\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1}(\mathbf{x} - \mathbf{m})$$

$p_{\mathcal{N}}$ is the density of the multi-variate normal distribution

Maximum Likelihood Update

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

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

independently of the given covariance matrix

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

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

$$\log p_{\mathcal{N}}(\mathbf{x} | \mathbf{m}, \mathbf{C}) = -\frac{1}{2} \log \det(2\pi \mathbf{C}) - \frac{1}{2} (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m})$$

$p_{\mathcal{N}}$ is the density of the multi-variate normal distribution

Variable Metric

On the function class

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

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

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

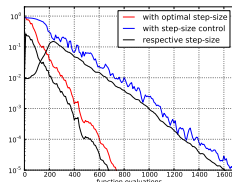
In effect, ellipsoidal level-sets are transformed into spherical level-sets.

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

On Convergence

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

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



Monte Carlo pure random search converges like

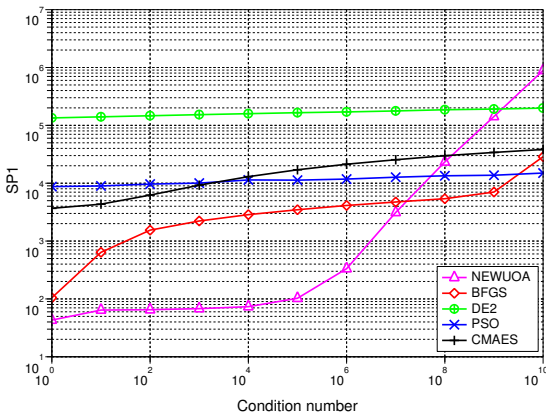
$$\|\mathbf{m}_k - \mathbf{x}^*\| \propto k^{-c} = e^{-c \log k}, \quad c = \frac{1}{n}$$

- 1 Problem Statement
- 2 Evolution Strategies
- 3 Step-Size Control
- 4 Covariance Matrix Adaptation
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments**
- 8 Summary and Final Remarks

Comparison to BFGS, NEWUOA, PSO and DE

f convex quadratic, separable with varying condition number α

Ellipsoid dimension 20, 21 trials, tolerance $1e-09$, eval max $1e+07$



BFGS (Broyden et al 1970)

NEWUOA (Powell 2004)

DE (Storn & Price 1996)

PSO (Kennedy & Eberhart 1995)

CMA-ES (Hansen & Ostermeier 2001)

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

\mathbf{H} diagonal

g identity (for **BFGS** and

NEWUOA)

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

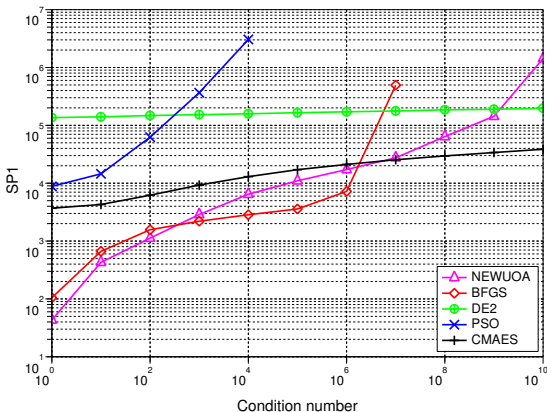
SP1 = average number of objective function evaluations¹⁴ to reach the target function value of $g^{-1}(10^{-9})$

¹⁴ 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

Rotated Ellipsoid dimension 20, 21 trials, tolerance $1e-09$, eval max $1e+07$



BFGS (Broyden et al 1970)

NEWUOA (Powell 2004)

DE (Storn & Price 1996)

PSO (Kennedy & Eberhart 1995)

CMA-ES (Hansen & Ostermeier 2001)

$f(x) = g(x^T H x)$ with

H full

g identity (for **BFGS** and **NEWUOA**)

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

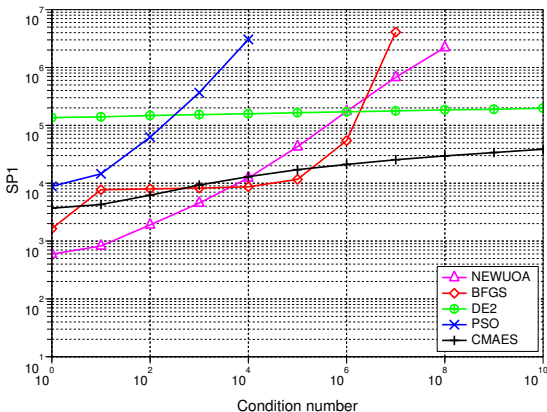
SP1 = average number of objective function evaluations¹⁵ to reach the target function value of $g^{-1}(10^{-9})$

¹⁵ 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

Sqrt of sqrt of rotated ellipsoid dimension 20, 21 trials, tolerance $1e-09$, eval max $1e+07$



BFGS (Broyden et al 1970)

NEWUOA (Powell 2004)

DE (Storn & Price 1996)

PSO (Kennedy & Eberhart 1995)

CMA-ES (Hansen & Ostermeier 2001)

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

\mathbf{H} full

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

NEWUOA)

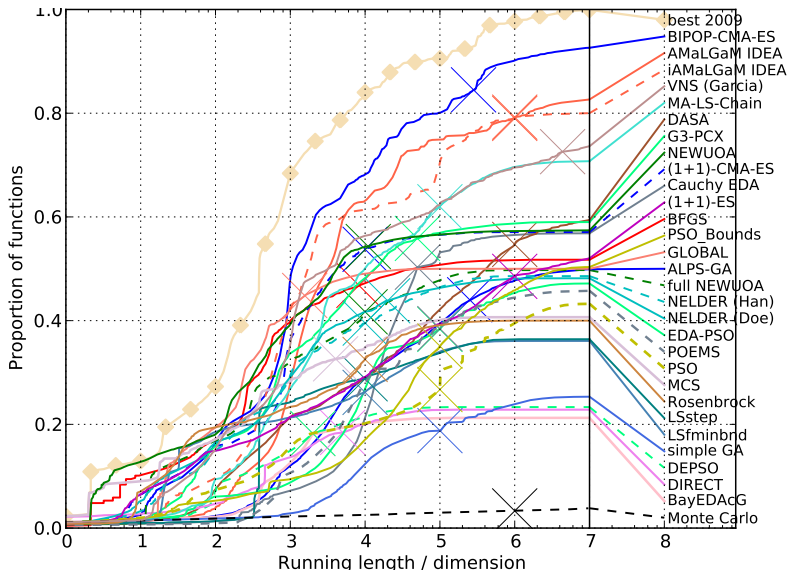
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations¹⁶ to reach the target function value of $g^{-1}(10^{-9})$

¹⁶ Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

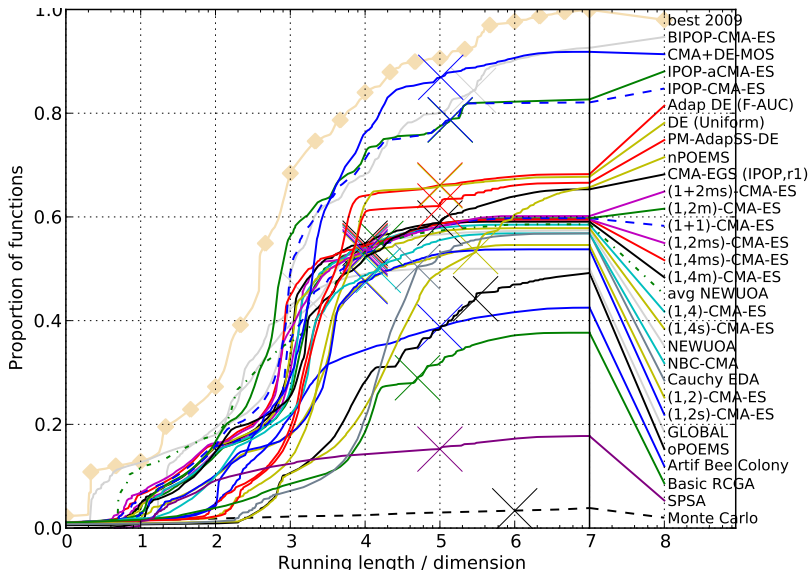
Comparison during BBOB at GECCO 2009

24 functions and 31 algorithms in 20-D



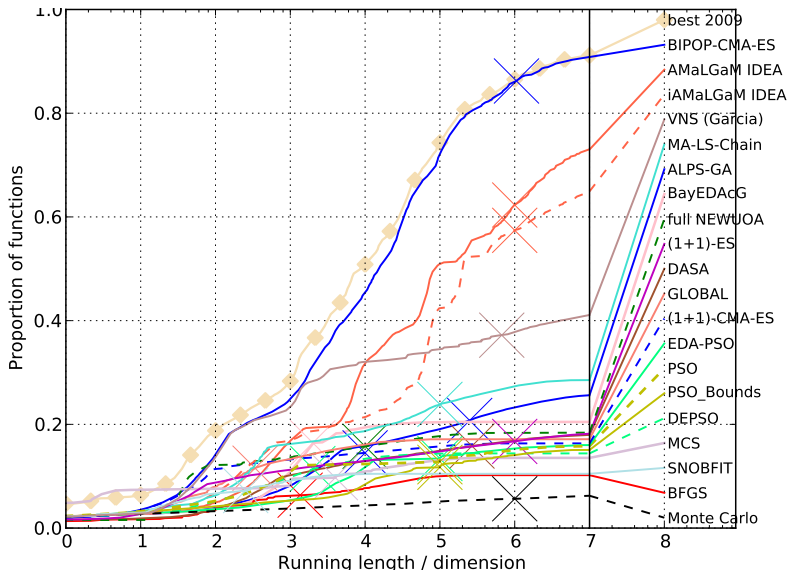
Comparison during BBOB at GECCO 2010

24 functions and 20+ algorithms in 20-D



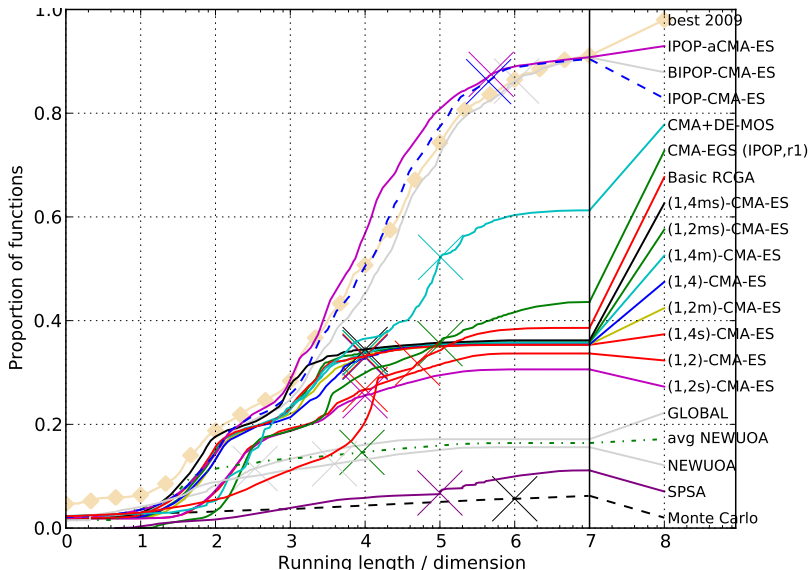
Comparison during BBOB at GECCO 2009

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



Comparison during BBOB at GECCO 2010

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



- 1 Problem Statement
- 2 Evolution Strategies
- 3 Step-Size Control
- 4 Covariance Matrix Adaptation
- 5 CMA-ES Summary
- 6 Theoretical Foundations
- 7 Comparing Experiments
- 8 Summary and Final Remarks**

The Continuous Search Problem

Difficulties of a non-linear optimization problem are

- dimensionality and non-separability

demands to exploit problem structure, e.g. neighborhood
cave: design of benchmark functions

- ill-conditioning

demands to acquire a second order model

- ruggedness

demands a non-local (stochastic? population based?) approach

Main Characteristics 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 $g(f(x))$ 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
in CMA-ES 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
 $\mathbf{C} \propto \mathbf{H}^{-1} \iff$ adapts a variable metric
 \iff new (rotated) problem representation
 $\implies f : \mathbf{x} \mapsto g(\mathbf{x}^T \mathbf{H} \mathbf{x})$ reduces to $\mathbf{x} \mapsto \mathbf{x}^T \mathbf{x}$

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 $< 100n$)
model-based methods

Thank You

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