Dynamic Problem Encoding for Optimization

Nikolaus Hansen

May 7, 2008

Content

Special thanks to
Anne Auger
Raymond Ros
Marc Schoenauer
Olivier Teytaud

Einstein once spoke of the "unreasonable effectiveness of mathematics" in describing how the natural world works. Whether one is talking about basic physics, about the increasingly important environmental sciences, or the transmission of disease, mathematics is never any more, or any less, than a way of thinking clearly. As such, it always has been and always will be a valuable tool, but only valuable when it is part of a larger arsenal embracing analytic experiments and, above all, wide-ranging imagination.

Lord Kay

Problem Statement: Search

Continuous Domain Search/Optimization

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

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

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

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

Problem Statement and Objectives

Continuous Domain Search/Optimization

- Goal
  - fast convergence toward the global optimum

    . . . or to a robust solution x
  - solution x with **small function value** with **least search cost**

    there are two (conflicting) objectives

- Typical Examples
  - shape optimization (e.g. using CFD) — curve fitting, airfoils
  - parameter calibration — controller, plants, images
  - model calibration — biological, physical

- Difficulties
  - exhaustive search is infeasible
  - deterministic search is often not successful
  - (naive) random search takes too long

Approach: stochastic search, Evolutionary Algorithms

. . . interface to real world problems

Problem Formulation

A real world problem requires

- a representation; the encoding of problem parameters into $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$
- the definition of a objective function $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ to be minimized

One might distinguish two approaches

Natural Encoding

Use a “natural” encoding and **design the optimizer** with respect to the problem e.g. use of specific “genetic operators”

frequently done in discrete domain

Concerned Encoding (Pure Black Box)

Use problem specific knowledge for encoding and use a **“generic” optimizer**

frequently done in continuous domain

Advantage: Sophisticated and well-validated optimizers can be used

How about *Adaptive Encoding*?

. . . function properties

Objective Function Properties

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

Additionally, f can be

- multimodal

  there are eventually many local optima

- non-smooth

  derivatives do not exist

- discontinuous
- ill-conditioned
- noisy
- ...

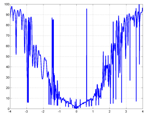
Goal : cope with any of these function properties

they are related to real-world problems

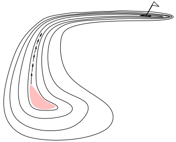
What Makes a Function Difficult to Solve?

Why stochastic search?

- ruggedness
  non-smooth, discontinuous, multimodal, and/or noisy function
- dimensionality
  (considerably) larger than three
- non-separability
  dependencies between the objective variables
- ill-conditioning



cut from 5-D solvable example



a narrow ridge

How Can a Difficult Function Be Solved?

…therefore…

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

Stochastic Search

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

Initialize distribution parameters $\boldsymbol{\theta}$, set sample size $\lambda \in \mathbb{N}$
While not terminate

1. **Sample distribution** $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. **Evaluate** $x_1, \ldots, x_\lambda$ **on** f
3. **Update parameters** $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

natural template for *Estimation of Distribution Algorithms*

Metaphors

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

. . . function properties

The Evolution Strategy

Minimize $f : \mathbb{R}^n \to \mathbb{R}$

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

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

- P is a **multi-variate normal** distribution

$$\mathcal{N}\left(\boldsymbol{m}_i, \sigma_i^2 \mathbf{C}_i\right) \sim \boxed{\boldsymbol{m}_i + \sigma_i \mathcal{N}(\mathbf{0}, \mathbf{C}_i)} \quad \text{for } i = 1, \ldots, \lambda$$

- $\boldsymbol{\theta} = \{\boldsymbol{m}_i, \mathbf{C}_i, \sigma_i\}_{i=1,\ldots,\lambda} \in (\mathbb{R}^n \times \mathbb{R}^{n \times n} \times \mathbb{R}_+)^\lambda$
- $F_\theta = F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$, where $\mu \leq \lambda$ and $\boldsymbol{x}_{i:\lambda}$ is the i-th best of the λ points

. . . why?

Why Normal Distributions?

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

   stable means the *sum* of normal variates is also normal, helpful in **design and analysis** of algorithms

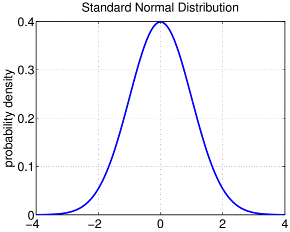
3. most convenient way to generate **isotropic** search points

   the isotropic distribution does **not favor any direction** (unfoundedly), supports rotational invariance

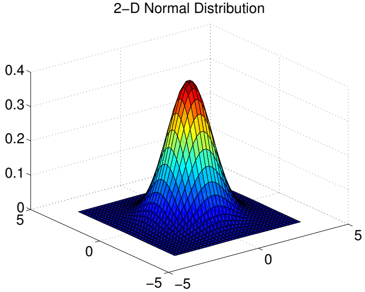
4. maximum entropy distribution with finite variance

   the least possible assumptions on f in the distribution shape

Normal Distribution



probability density of 1-D standard normal distribution



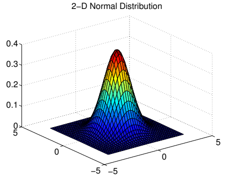
probability density of 2-D normal distribution

The Multi-Variate (n-Dimensional) Normal Distribution

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

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

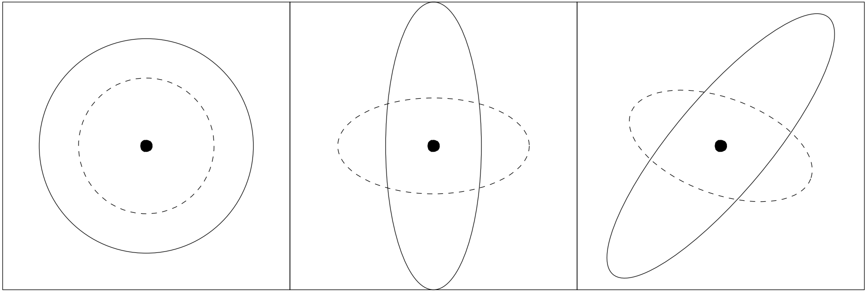
- determines the displacement (translation)
- is the value with the largest density (modal value)
- the distribution is symmetric about the distribution mean



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

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

Lines of Equal Density



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

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

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

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

...CMA

Sampling New Search Points

The governing equation for derandomized Evolution Strategies

New search points are sampled normally distributed

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

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

where

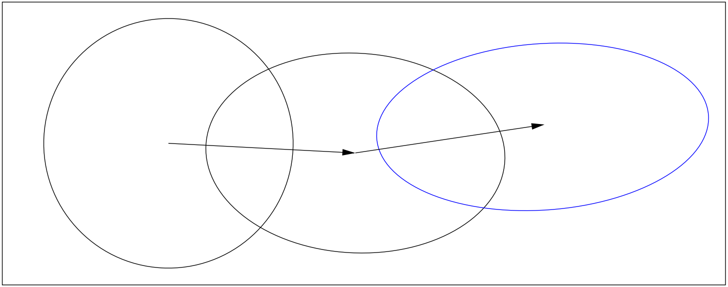
- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the **shape** of the distribution ellipsoid

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

Covariance Matrix Adaptation

Rank-One Update

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



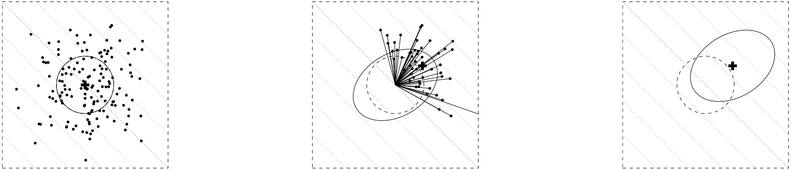
new distribution,

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

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

Rank-μ Update

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



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

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

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

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

calculating $\mathbf{C}$ from $\mu = 50$ points, $w_1 = \cdots = w_\mu = \frac{1}{\mu}$

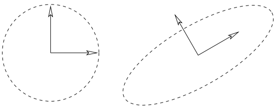
new distribution

Remark: the old (sample) distribution shape has a great influence on the new distribution $\longrightarrow$ iterations needed

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

covariance matrix adaptation in the evolution strategy

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



- learns a new, **rotated problem representation** and a **new metric** (Mahalanobis)
  components are independent (only) in the new representation
- approximates the inverse Hessian on quadratic functions
  overwhelming empirical evidence, proof is in progress

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

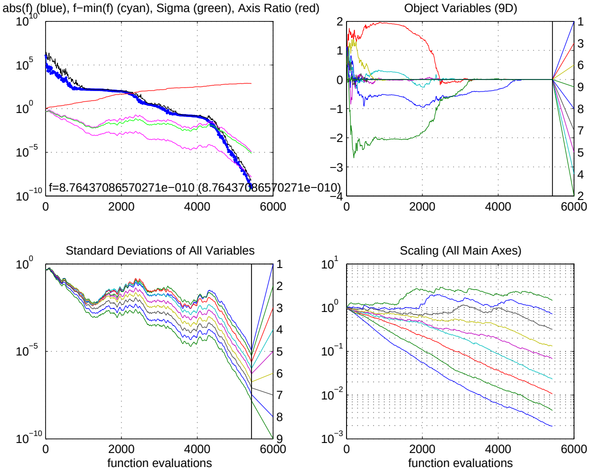
covariance matrix adaptation

- is equivalent with an adaptive (general) linear encoding[1]

[1] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

Experimentum Crucis (1)

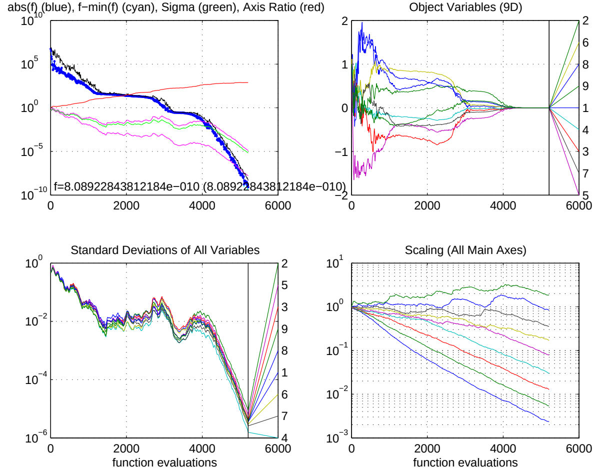
f convex quadratic, separable



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

Experimentum Crucis (2)

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



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

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

Invariance

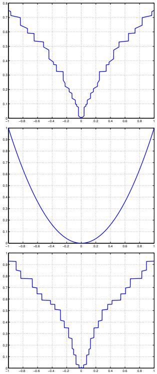
The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.

— Albert Einstein

Invariance is a *guaranty for generalization* of performance from a single function to a class of functions.
Most important invariance properties of the Covariance Matrix Adaptation (CMA) Evolution Strategy (ES) are

- invariance to **order preserving transformations** in function space
- Translation and **rotation invariance** in search space

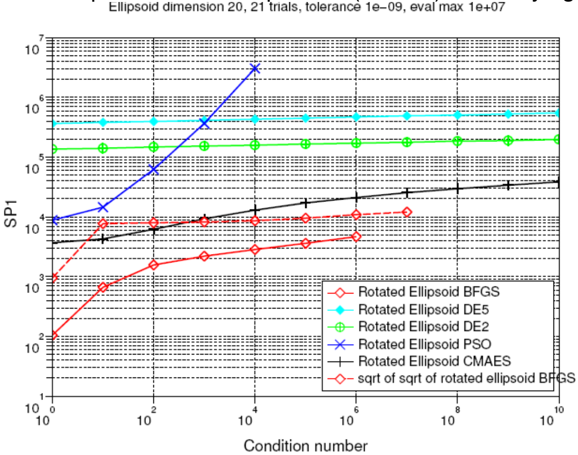
to *rigid transformations* of the search space



. . . empirical validation

Comparison to , BFGS, PSO and DE

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



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

g identity (BFGS, *red*) or

$g(.) = (.)^{1/4}$ (BFGS, *red dashed*) or

g order-preserving = strictly increasing (all other)

BFGS: quasi-Newton method

PSO: Particle Swarm Optimization

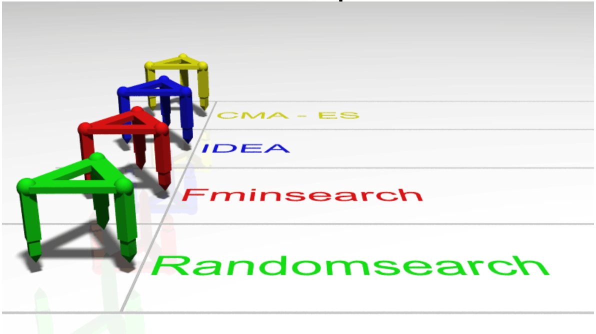
DE:Differential Evolution

CMA-ES —

SP1 = average number of objective function evaluations to reach the target function value of 10^{-9}

. . . population size, invariance

Comparison to IDEA and Simplex-Downhill



CMA-ES: Covariance Matrix Adaptation Evolution Strategy[2]
IDEA: Iterated Density-Estimation Evolutionary Algorithm[3]
Fminsearch: Nelder-Mead simplex downhill method[4]

Randomsearch: pure Monte-Carlo sampling

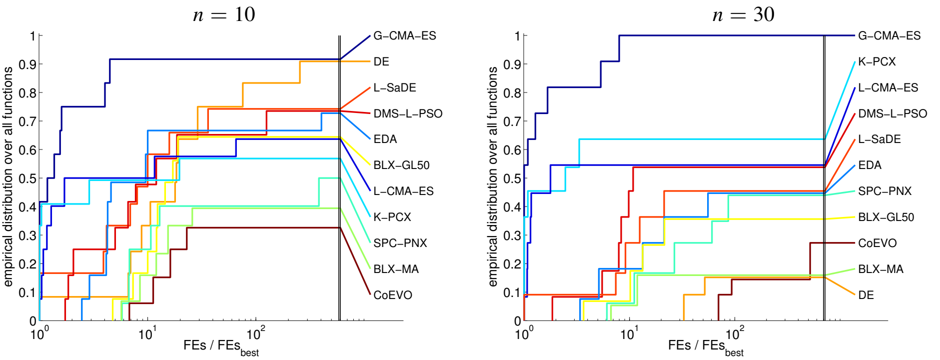
Peter Dürr and Andreas Pfister 2004. Optimization of Walking Gaits for a Three Legged Walking Robot, Diploma Thesis, Institut für Mechanische Systeme, ETH Zurich

---

[2] Hansen (2001) Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation Journal*

CEC 2005 Real Parameter Optimization Session

Empirical Distribution of Normalized Success Performance



$\texttt{FEs} = \texttt{mean}(\#fevals) \times \frac{\#\text{all runs (25)}}{\#\text{successful runs}}$, where $\#fevals$ includes only successful runs.

Shown: **empirical distribution function** of the Success Performance $\texttt{FEs}$ divided by $\texttt{FEs}$ of the best algorithm on the respective function.

Results of all functions are used where at least one algorithm was successful at least once, i.e. where the target function value was reached in at least one experiment (out of 11×25 experiments).

Small values for $\texttt{FEs}$ and therefore large (cumulative frequency) values in the graphs are preferable.

The Covariance Matrix Adaptation Evolution Strategy

In a Nutshell

1. Multivariate normal distribution to generate new search points

   follows the maximum entropy principle

2. Selection only based on the ranking of the f-values, weighted recombination

   using only the ranking of f-values preserves invariance

3. *Covariance matrix adaptation (CMA)* **increases the probability** to repeat **successful steps**

   learning all pairwise dependencies
   $\Longrightarrow$ conducts an incremental PCA
   $\Longrightarrow$ new (rotated) problem representation

4. An **evolution path** (a trajectory) is exploited in two places
   - enhances the covariance matrix (rank-one) adaptation

     yields sometimes linear time complexity
   - controls the **step-size** (step length)

     aims at conjugate perpendicularity

Linear Encoding and the Covariance Matrix

Equivalence between change in encoding and transformation of the mutation operator

Let $\boldsymbol{x}_B, \boldsymbol{x}_A \in \mathbb{R}^n$ be **two genotypes** encoding the same phenotype

$$\boldsymbol{y} = \mathbf{A}\,\boldsymbol{x}_A = \mathbf{B}\,\boldsymbol{x}_B$$

The **effect of the different encodings** becomes evident, when the genotype is changed (adding $\mathcal{N}(\mathbf{0}, \mathbf{C})$).

$$\begin{aligned} \boldsymbol{y}_{\text{new}} &= \mathbf{B}\,(\boldsymbol{x}_B + \mathcal{N}(\mathbf{0}, \mathbf{C})) &&= \mathbf{B}\,\boldsymbol{x}_B + \mathbf{B}\,\mathcal{N}(\mathbf{0}, \mathbf{C}) \\ &&&= \mathbf{A}\,\boldsymbol{x}_A + \mathbf{A}\,\mathbf{A}^{-1}\mathbf{B}\,\mathcal{N}(\mathbf{0}, \mathbf{C}) \\ &&&= \mathbf{A}\,(\boldsymbol{x}_A + \mathbf{A}^{-1}\mathbf{B}\,\mathcal{N}(\mathbf{0}, \mathbf{C})) \\ \boldsymbol{y}_{\text{new}} &= \mathbf{A}\,(\boldsymbol{x}_A + \mathbf{A}^{-1}\mathbf{B}\,\mathcal{N}(\mathbf{0}, \mathbf{C})) \end{aligned}$$

Using a new encoding $\mathbf{B}$ means using a different covariance matrix

Adaptive Encoding

Definition (Adaptive Encoding)

Given a search algorithm, $\mathcal{A}$ in state s, an encoding, T_B and an update, $\mathcal{U}$, then the iteration step

$$s \leftarrow T_B \circ \mathcal{A} \circ T_B^{-1}(s) \quad (1)$$

$$\mathbf{B} \leftarrow \mathcal{U}(\mathbf{B}, s) \quad (2)$$

defines an *adaptive encoding* where $T_B \circ \mathcal{A} \circ T_B^{-1}(s) = T_B(\mathcal{A}(T_B^{-1}(s)))$.

Remark (Evaluation of Solutions)

In order to make use of Eq. (1), $\mathcal{A}$ has to operate on $f \circ \mathbf{B}$.

Adaptive Encoding

Example: Adaptive Encoding of CSA-ES

AE_{CMA}-CSA-ES

1 initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean), $\boldsymbol{p}_\sigma = \mathbf{0}$ (evolution path), $\sigma > 0$ (step-size)
2 initialize $\boldsymbol{B} = \boldsymbol{B}^\circ = \boldsymbol{I}$ (encoding matrices)
3 **repeat**
4 $\quad \boldsymbol{m} \leftarrow \boldsymbol{B}^{-1}\boldsymbol{m}$
5 $\quad \boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^{\circ\mathrm{T}}\boldsymbol{p}_\sigma$
6 $\quad$ **begin**
7 $\quad\quad \boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\mathcal{N}_i\left(\mathbf{0}, \boldsymbol{I}\right), \quad \text{for } i = 1, \ldots, \lambda$
8 $\quad\quad f_i = f \circ \boldsymbol{B}\left(\boldsymbol{x}_i\right) = f(\boldsymbol{B}\,\boldsymbol{x}_i), \ \text{for } i = 1, \ldots, \lambda$ `// encode to evaluate`
9 $\quad\quad \boldsymbol{m}^- = \boldsymbol{m}$
10 $\quad\quad \boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i(f)}$
11 $\quad\quad \boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2 - c_\sigma)\mu_{\mathrm{w}}}\,\frac{1}{\sigma}(\boldsymbol{m} - \boldsymbol{m}^-)$
12 $\quad\quad \sigma \leftarrow \sigma\,\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\mathbf{0}, \boldsymbol{I})\|} - 1\right)\right)$
13 $\quad$ **end**
14 $\quad \boldsymbol{m} \leftarrow \boldsymbol{B}\boldsymbol{m}$
15 $\quad \boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^\circ\boldsymbol{p}_\sigma$
16 $\quad \text{AE}_{\text{CMA}}\text{-Update}(\{\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu\})$ `// update` $\boldsymbol{B}$ `and` $\boldsymbol{B}^\circ$
17 **until** *stopping criterion is met*

AE : Adaptive Encoding
CMA : Covariance Matrix Adaptation
CSA-ES : Cumulative *Step-size Adaptation* Evolution Strategy, lines 6–13

Adaptive Encoding

Theorem (Recovery of CMA-ES)

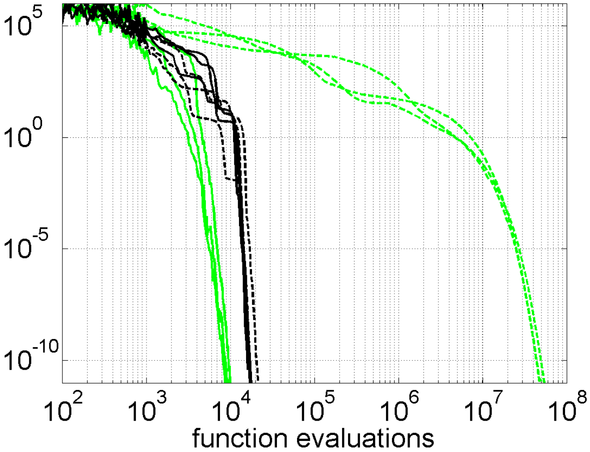
Given AE_{CMA}-Update in Procedure 1, the AE_{CMA}-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES implements the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES.

Adaptive Encoding

- can render *any* continuous domain search algorithm independent of the coordinate system
- anticipated successful applications in particular for population-based stochastic algorithms

Another Case Study

Adaptive Encoding

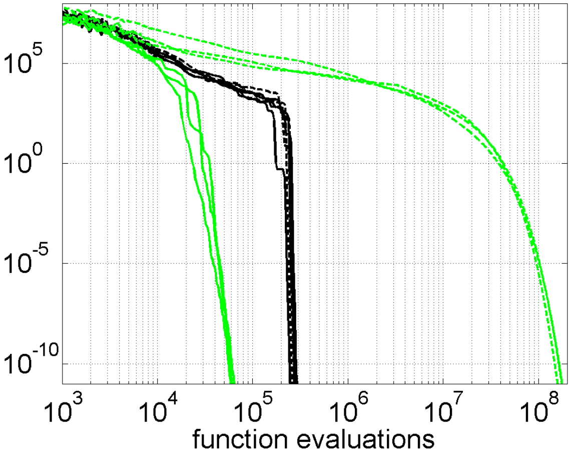


Cauchy-ES (green) versus Adaptively Encoded Cauchy-ES (black)

rotating a separable function, 10-D

Another Case Study

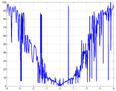
Adaptive Encoding

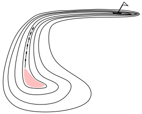


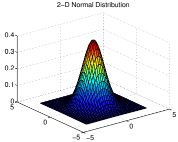
Cauchy-ES (green) versus Adaptively Encoded Cauchy-ES (black)

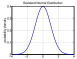
rotating a separable function, 30-D

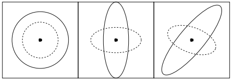

x
f(x)

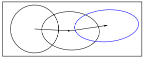


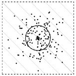



2–D Normal Distribution

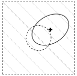

Standard Normal Distribution

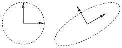


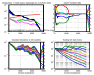


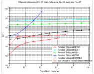


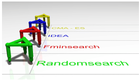
Thank You

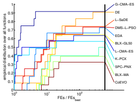


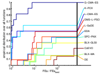


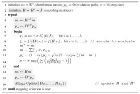


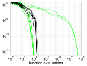


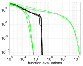

IDEA
Fminsearch
Randomsearch


G–CMA–ES
DE
L–SaDE
DMS–L–PSO
EDA
BLX–GL50
L–CMA–ES
K–PCX
SPC–PNX
BLX–MA
CoEVO






function evaluations


function evaluations