Introduction to Randomized Continuous Optimization

Youhei Akimoto[1] & Anne Auger[2] & Nikolaus Hansen[2]
1. Shinshu University, Nagano, Japan
2. Inria, Research Centre Saclay, France

y_akimoto@shinshu-u.ac.jp
anne.auger@lri.fr
nikolaus.hansen@lri.fr

http://www.sigevo.org/gecco-2016/

GECCO'16 Companion, July 20–24, 2016, Denver, CO, USA.
ACM 978-1-4503-4323-7/16/07.
http://dx.doi.org/10.1145/2908961.2926980

Denver, Colorado, USA
July 20-24, 2016

We are happy to answer questions at any time.

Overview

Problem Statement

Continuous Domain Search/Optimization

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

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

- Black Box scenario (direct search scenario)

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

Problem Statement

Continuous Domain Search/Optimization

- Goal
  - fast convergence to the global optimum
  - 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) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images
- Problems
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement | Black Box Optimization and Its Difficulties

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) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images
- Problems
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement | Black Box Optimization and Its Difficulties

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) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images
- Problems
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Objective Function Properties

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

Additionally, f can be

- non-convex
- multimodal
  there are possibly many local optima
- non-smooth
  derivatives do not exist
- discontinuous, plateaus
- ill-conditioned
- noisy
- ...

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

Objective Function Properties

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

- non-convex
- multimodal

  there are possibly many local optima
- non-smooth

  derivatives do not exist
- discontinuous, 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?

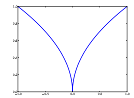
- non-linear, non-quadratic, non-convex

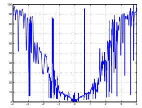
  on linear and quadratic functions much better search policies are available
- ruggedness

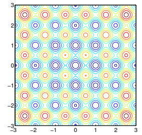
  non-smooth, discontinuous, multimodal, and/or noisy function
- dimensionality (size of search space)

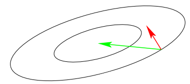
  (considerably) larger than three
- non-separability

  dependencies between the objective variables
- ill-conditioning



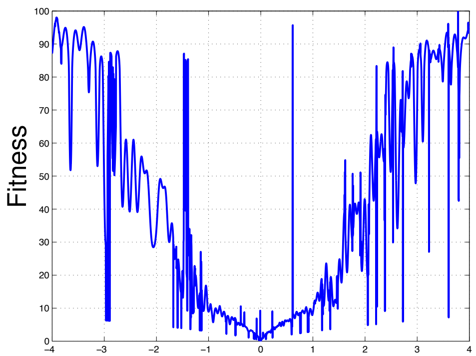






Ruggedness

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 20 points equally spaced onto the interval $[0, 1]$. Now consider the 10-dimensional space $[0, 1]^{10}$. To get similar coverage in terms of distance between adjacent points requires $20^{10} \approx 10^{13}$ points. 20 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 that is valuable in small dimensions might be useless in moderate or large dimensional search spaces. Example: exhaustive search.

Separable Problems

Definition (Separable Problem)

A function f is separable if

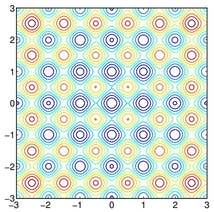
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

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

Example: Additively decomposable functions

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

Rastrigin function



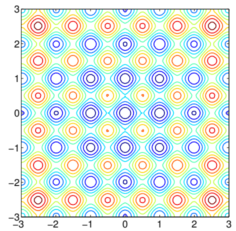
Non-Separable Problems

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

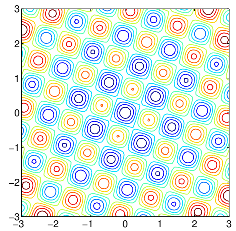
Rotating the coordinate system

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

$\mathbf{R}$ rotation matrix



$\xrightarrow{\mathbf{R}}$



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

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

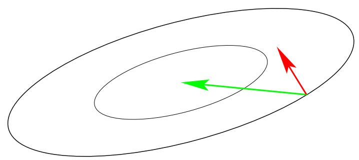
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

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

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



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

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

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

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

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	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 Programmin
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

Landscape of Continuous Black-Box Optimization

Deterministic algorithms

Quasi-Newton with estimation of gradient (BFGS) [Broyden et al. 1970]
Simplex downhill [Nelder & Mead 1965]
Pattern search [Hooke and Jeeves 1961]
Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]

Stochastic (randomized) search methods

Evolutionary Algorithms (continuous domain)
- Differential Evolution [Storn & Price 1997]
- Particle Swarm Optimization [Kennedy & Eberhart 1995]
- **Evolution Strategies, CMA-ES** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Estimation of Distribution Algorithms (EDAs) [Larrañaga, Lozano, 2002]
- Cross Entropy Method (same as EDA) [Rubinstein, Kroese, 2004]
- Genetic Algorithms [Holland 1975, Goldberg 1989]

Simulated annealing [Kirkpatrick et al. 1983]
Simultaneous perturbation stochastic approximation (SPSA) [Spall 2000]

Overview

Stochastic Search

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

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate
1. Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \dots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\boldsymbol{x}_1, \dots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \dots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

Natural template for (incremental) *Estimation of Distribution Algorithms*

Examples

1. Estimation of Distribution Algorithms
2. Evolution Strategies
3. Differential Evolution
4. Particle Swarm Optimization

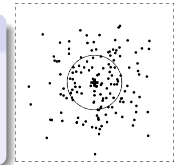
all those methods are comparison based

Evolution Strategies

New search points are sampled normally distributed

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

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



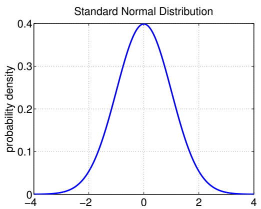
where

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

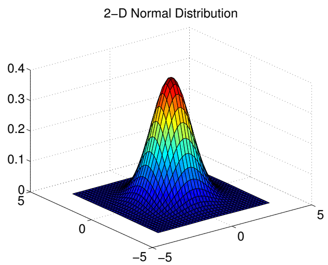
here, all new points are sampled with the same parameters

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

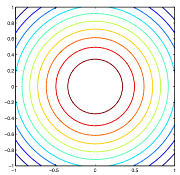
Normal Distribution



probability density of the 1-D standard normal distribution



probability density of a 2-D normal distribution

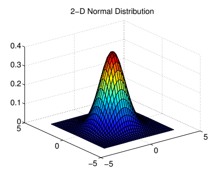


The Multi-Variate (n-Dimensional) Normal Distribution

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

The mean value $\boldsymbol{m}$

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

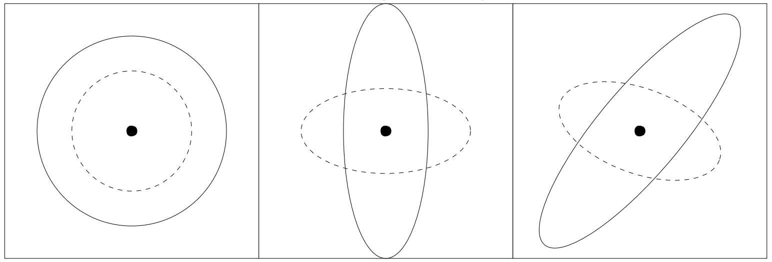


The covariance matrix $\mathbf{C}$

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

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

Lines of Equal Density



$\mathcal{N}(m, \sigma^2 \mathbf{I}) \sim m + \sigma \mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(m, \mathbf{D}^2) \sim m + \mathbf{D} \mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(m, \mathbf{C}) \sim m + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
one degree of freedom σ	n degrees of freedom	$(n^2 + n)/2$ degrees of freedom
components are independent standard normally distributed	components are independent, scaled	components are correlated

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

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

Non-elitist selection and intermediate (weighted) recombination

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

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

$$m \leftarrow \sum_{i=1}^{\mu} w_i x_{i:\lambda} = m + \sigma \underbrace{\sum_{i=1}^{\mu} w_i y_{i:\lambda}}_{=: y_w}$$

where

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

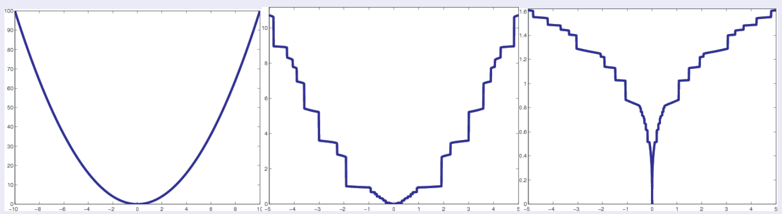
The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

Invariance Under Monotonically Increasing Functions

Rank-based algorithms

Update of all parameters uses only the ranks

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



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

g is strictly monotonically increasing
g preserves ranks

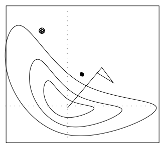
3

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

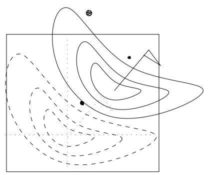
Basic Invariance in Search Space

- translation invariance

is true for most optimization algorithms



$f(x) \leftrightarrow f(x - a)$



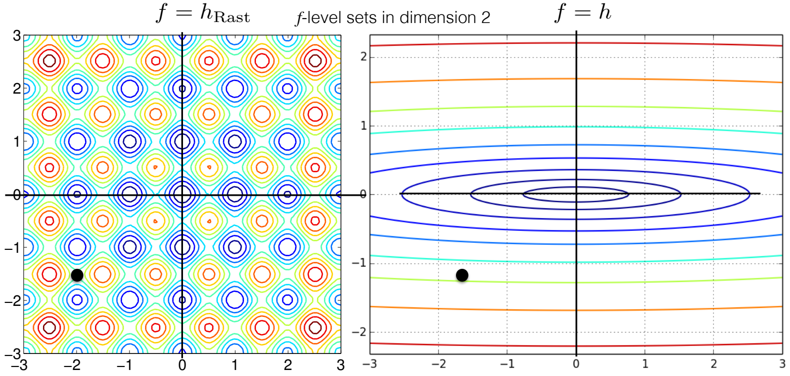
Identical behavior on f and f_a

$$f: \quad x \mapsto f(x), \qquad x^{(t=0)} = x_0$$
$$f_a: \quad x \mapsto f(x - a), \quad x^{(t=0)} = x_0 + a$$

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

Invariance Under Rigid Search Space Transformations

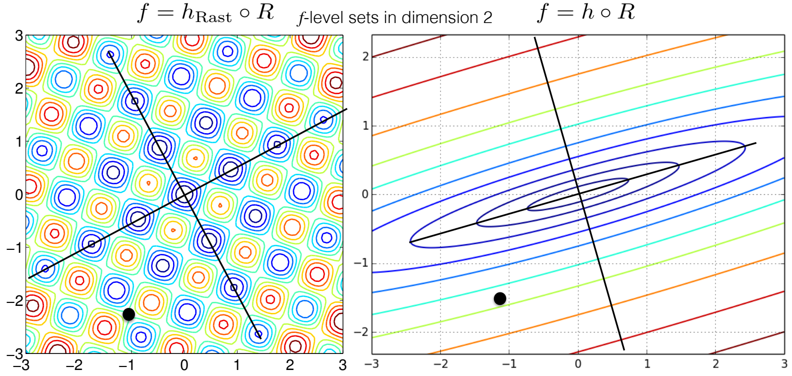
$f = h_{\text{Rast}}$ f-level sets in dimension 2 $f = h$



for example, invariance under search space rotation
(separable ⇔ non-separable)

Invariance Under Rigid Search Space Transformations

$f = h_{\text{Rast}} \circ R$ f-level sets in dimension 2 $f = h \circ R$



for example, invariance under search space rotation
(separable ⇔ non-separable)

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
  - from benchmark functions
  - from solved real world problems

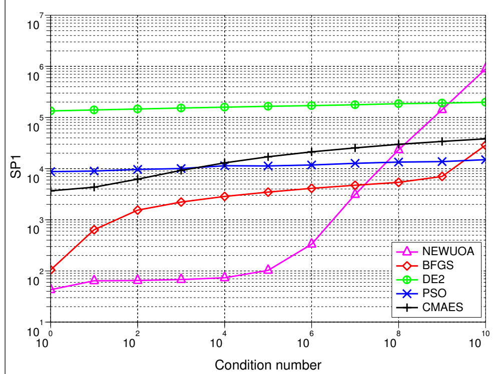
  are only useful if they do generalize to other problems
- Invariance is a strong non-empirical statement about generalization
  generalizing (identical) performance from a single function to a whole class of functions

consequently, invariance is important for the evaluation of search algorithms

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

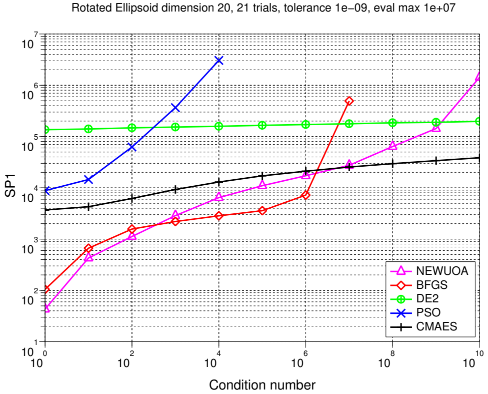
$f(\boldsymbol{x}) = g(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x})$ with
$\boldsymbol{H}$ diagonal
g identity (for BFGS and NEWUOA)
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[14] 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



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

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

$\boldsymbol{H}$ full

g identity (for BFGS and NEWUOA)

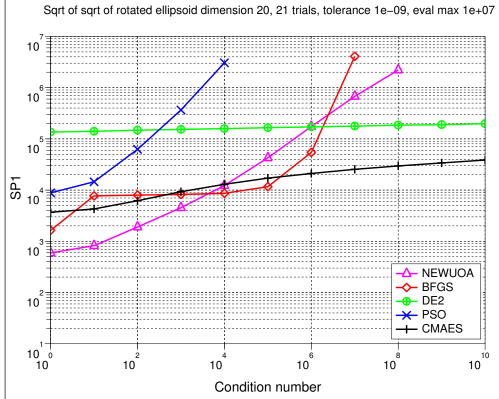
g any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations[15] 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



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

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

$\boldsymbol{H}$ full

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

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

SP1 = average number of objective function evaluations[16] to reach the target function value of $g^{-1}(10^{-9})$

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

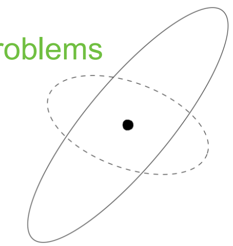
Overview

Zoom on ESs: Objectives

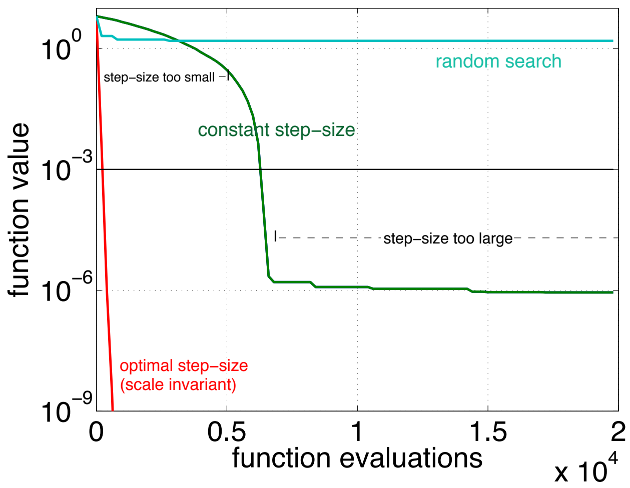
Illustrate why and how sampling distribution is controlled

step-size control (overall standard deviation)
allows to achieve linear convergence

covariance matrix control
allows to solve ill-conditioned problems



Why Step-Size Control?



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

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

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

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

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

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

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

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

[e] Ostermeier *et al* 1994, Step-size adaptation based on non-local use of selection information, *PPSN IV*

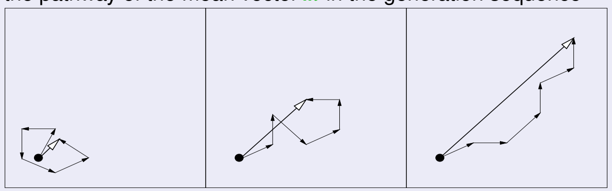
Path Length Control (CSA)

The Concept of Cumulative Step-Size Adaptation

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{m} + \sigma\, \boldsymbol{y}_i \\ \boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \end{aligned}$$

Measure the length of the *evolution path*

the pathway of the mean vector $\boldsymbol{m}$ in the generation sequence



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

loosely speaking steps are

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

Path Length Control (CSA)

The Equations

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

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

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

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

Path Length Control (CSA)

The Equations

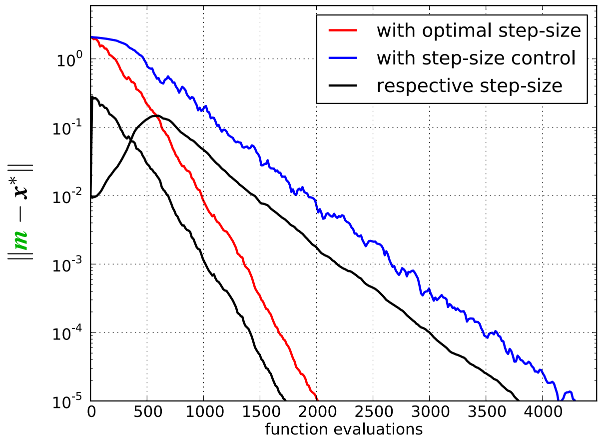
Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

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

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

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

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



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

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

Overview

❶ Problem Statement
- Continuous Black-Box Optimization
- Typical Difficulties

❷ Stochastic Black-Box Algorithms
- General Template
- Invariance
- Comparisons of a few DFOs

❸ Zoom on Evolution Strategies
- Step-size Adaptation
- Covariance Matrix Adaptation

❹ Evaluating Black-Box Algorithms
- Displaying results and visualization
- Statistics
- Average Runtime
- Empirical Cumulative Distribution Function (ECDF)

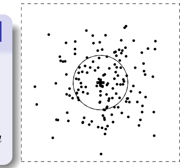
Evolution Strategies

Recalling

New search points are sampled normally distributed

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

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



where

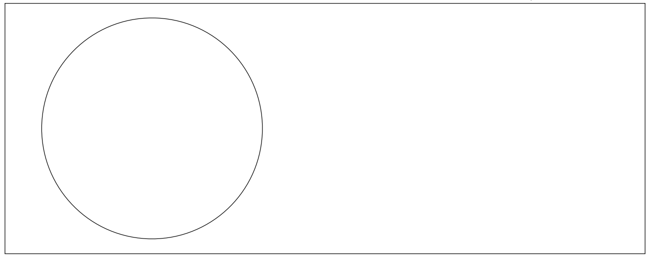
- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$ determines the shape of the distribution ellipsoid

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

Covariance Matrix Adaptation

Rank-One Update

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



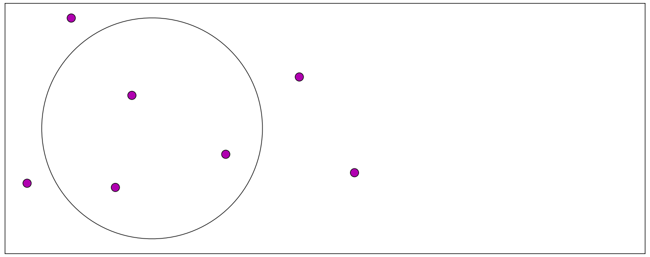
initial distribution, $\mathbf{C} = \mathbf{I}$

... equations

Covariance Matrix Adaptation

Rank-One Update

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



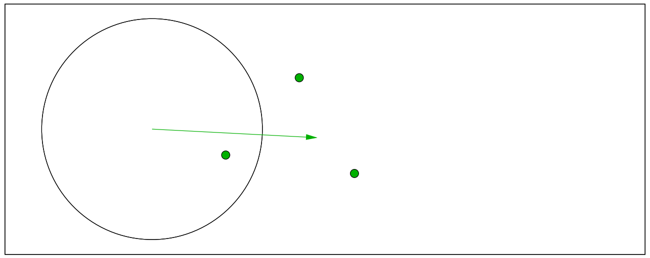
initial distribution, $\mathbf{C} = \mathbf{I}$

... equations

Covariance Matrix Adaptation

Rank-One Update

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



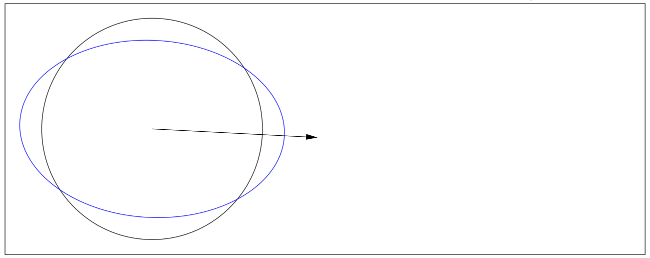
$\boldsymbol{y}_w$, movement of the population mean $\boldsymbol{m}$ (disregarding σ)

... equations

Covariance Matrix Adaptation

Rank-One Update

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



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

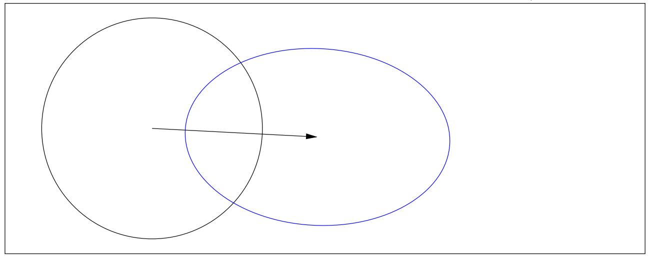
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

... equations

Covariance Matrix Adaptation

Rank-One Update

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



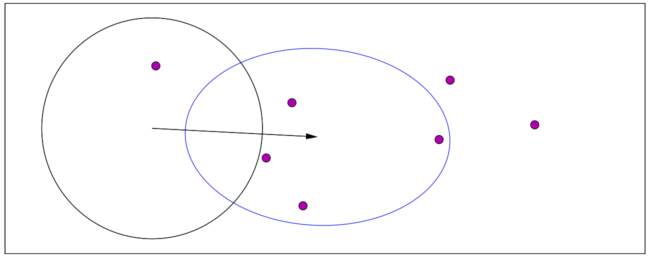
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



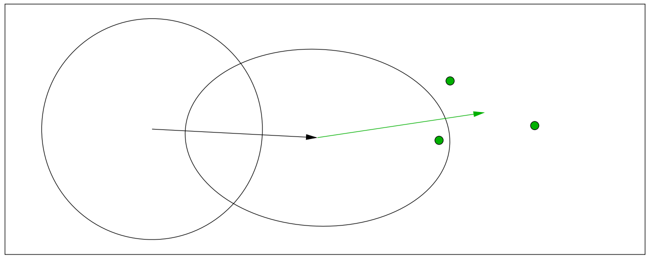
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



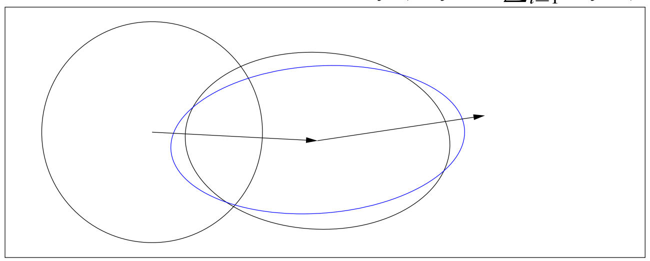
movement of the population mean m

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



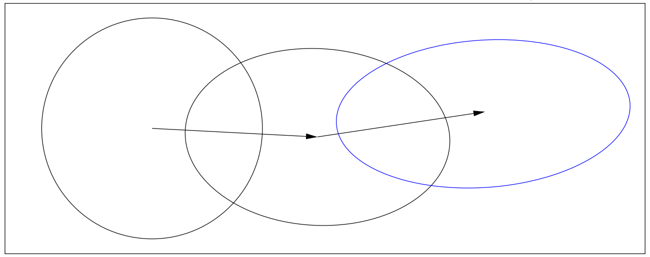
mixture of distribution $\mathbf{C}$ and step y_w,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times y_w y_w^{\mathrm{T}}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



new distribution,

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

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

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

…equations

Covariance Matrix Adaptation

Rank-One Update

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

While not terminate

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

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

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

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

The CMA-ES

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

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

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 $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \approx 0.3\,\lambda$

While not terminate

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

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

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

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

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

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

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

Experimentum Crucis (0)

What did we want to achieve?

- reduce any convex-quadratic function

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

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

to the sphere model

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

without use of derivatives

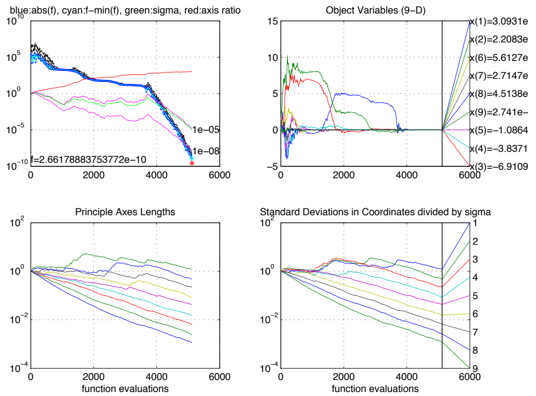
- lines of equal density align with lines of equal fitness

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

in a stochastic sense

Experimentum Crucis (1)

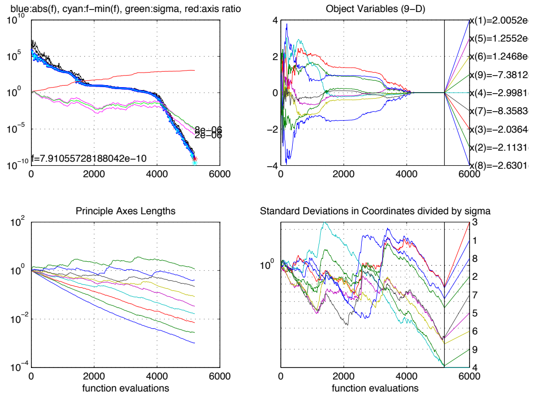
f convex quadratic, separable



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

Experimentum Crucis (2)

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



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

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

Overview

Evaluation of Anytime Black-Box Optimizers

Particularly Randomized Search Algorithms

Randomized optimization is mostly an empirical science

Hence it is crucial to properly conduct numerical experiments and assess performance

to not fool ourself on what our favorite algorithm is good/not good at

in order to not fool others …

"The first principle is that you must not fool yourself and you are the easiest person to fool."
Richard P. Feynman

Evaluation of Anytime Black-Box Optimizers

Particularly Randomized Search Algorithms

Evaluation of performance in a broad sense

not only be able to say “Algorithm A is better than B”

but

understand where and why algorithm work

quantify performance

Measuring Performance

Empirically

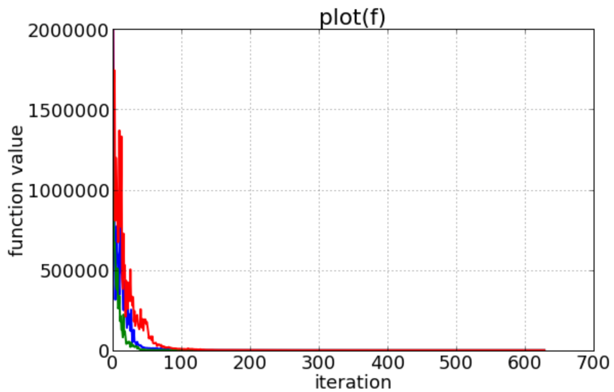
convergence graphs is all we have to start with

having the right presentation is important

too often neglected

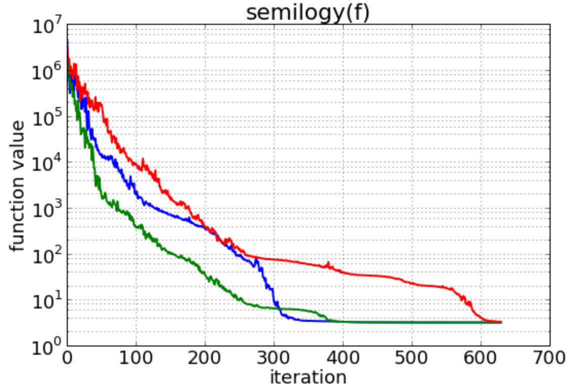
the details are important

Displaying Three Runs



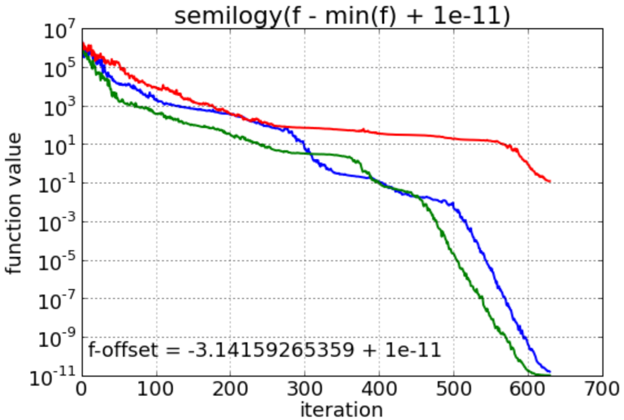
not like this (it's unfortunately a common picture)

Displaying Three Runs



better like this (shown are the same data),
caveat: fails with negative f-values

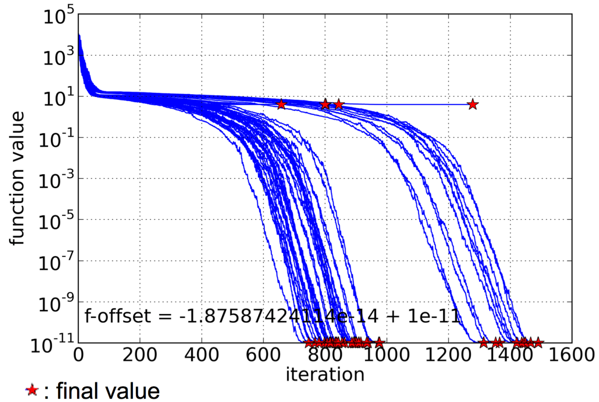
Displaying Three Runs



even better like this: subtract minimum value over all runs

Displaying 51 Runs

don't hesitate to display all data (the appendix is your friend)



★ : final value

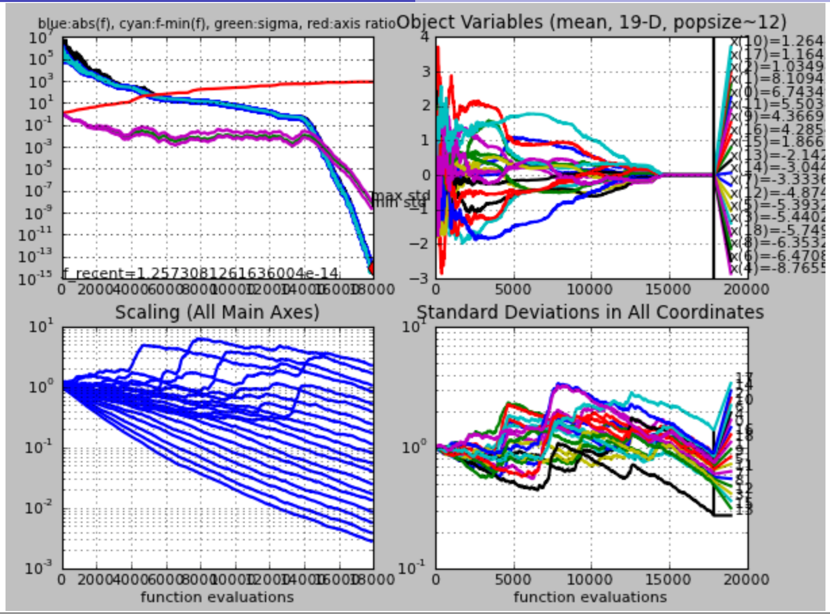
observation: three different "modes", which would be difficult to represent or recover in single statistics

Visualization

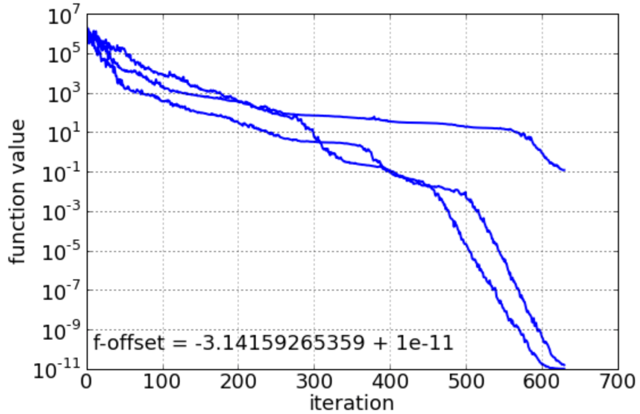
Other data than convergence graphs can be very instructive to visualize to understand performance

Example:

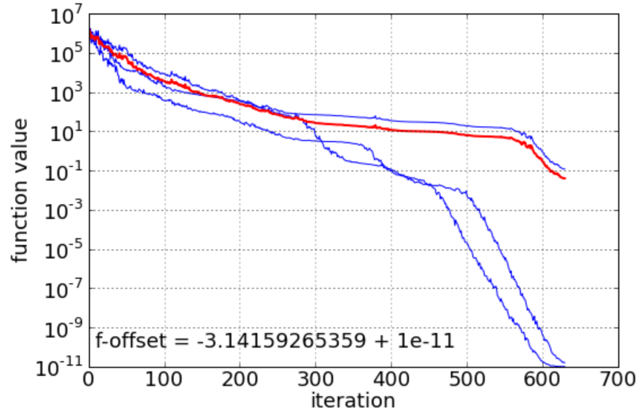
visualization of evolution of distribution in CMA-ES (see next slide)



Which Statistics?



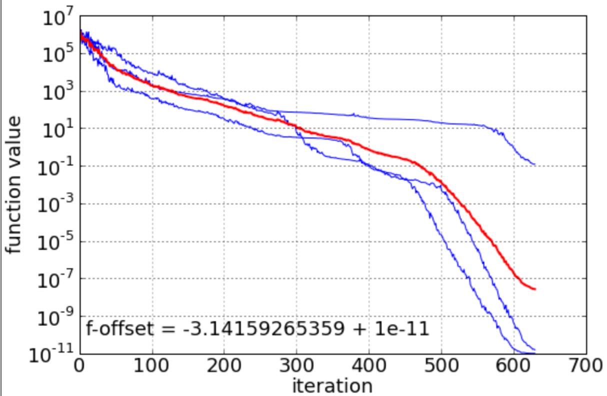
Which Statistics?



mean/average function value

- tends to emphasize large values

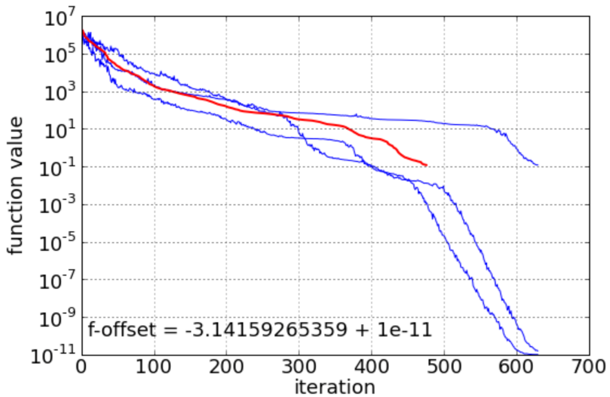
Which Statistics?



geometric average function value $\exp(\mathrm{mean_i}(\log(f_i))) = (\prod_{i=1}^{N} f_i)^{1/N}$

- reflects "visual" average
- depends on offset
- artefact due to adding 1e-11

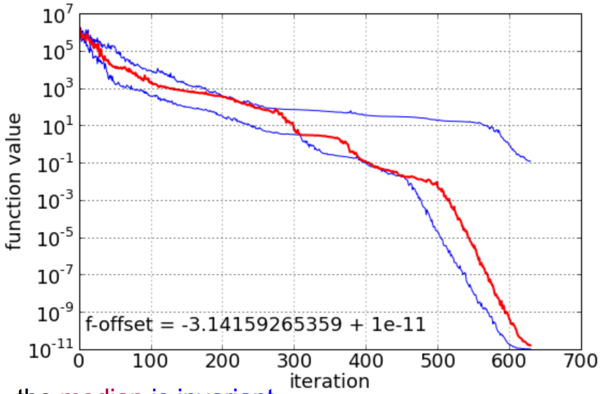
Which Statistics?



average iterations

- reflects "visual" average
- here: incomplete

Which Statistics?



the median is invariant

- unique for uneven number of data
- independent of log-scale, offset...

  median(log(data))=log(median(data))
- same when taken over x- or y-direction

Implication

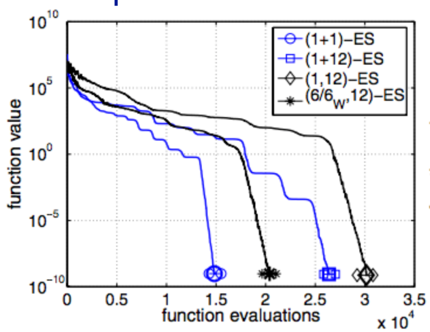
use the median as summary datum

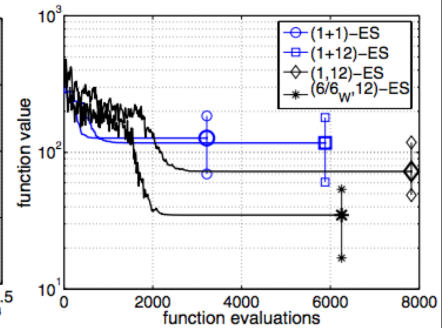
more general: use quantiles as summary data

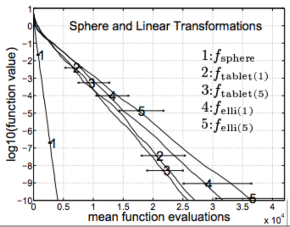
for example out of 15 data: 2nd, 8th, and 14th value represent the 10%, 50%, and 90%-tile

unless there are good reasons for a different statistics

Examples







Comparison of 4 algorithms using the "median run" and the 90% central range of the final value on two different functions (Ellipsoid and Rastrigin)

caveat: this range display with simple error bars fails, if, e.g., 30% of all runs "converge"

Statistical Assessment

❶ Assess the meaning/relevance of a difference first (the only difficult part)

using enough data, any difference can be made significant

Statistical Assessment

❷ Apply rank-sum test (Wilcoxon, Mann-Whitney U)
only assumption: no equal data values

hypothesis:

compares s$\Pr(x > y) \neq \Pr(x < y) \neq 1/2$anking

two-sided 1%-significance p-value needs only 2x5 data values

For the same p-value, fewer significant data are better
using enough data, any difference can be made significant

Generally: non-parametric tests, Kolmogorov-Smirnov test for ECDFs, no need to use the t-test

Overview

❶ Problem Statement
- Continuous Black-Box Optimization
- Typical Difficulties

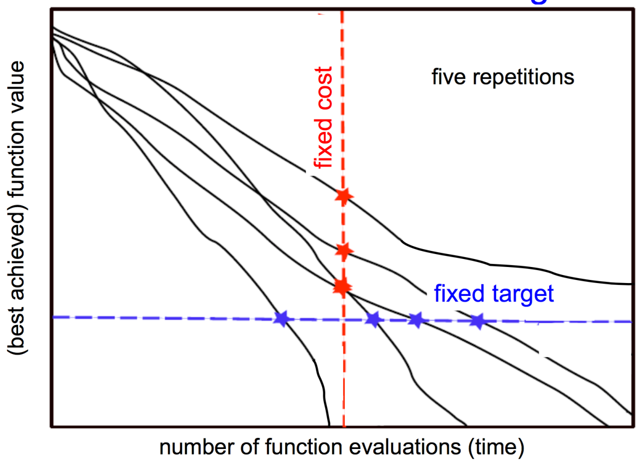
❷ Stochastic Black-Box Algorithms
- General Template
- Invariance
- Comparisons of a few DFOs

❸ Zoom on Evolution Strategies
- Step-size Adaptation
- Covariance Matrix Adaptation

❹ Evaluating Black-Box Algorithms
- Displaying results and visualization
- Statistics
- Average Runtime
- Empirical Cumulative Distribution Function (ECDF)

Measuring Performance from Convergence Graphs

fixed-cost versus fixed-target



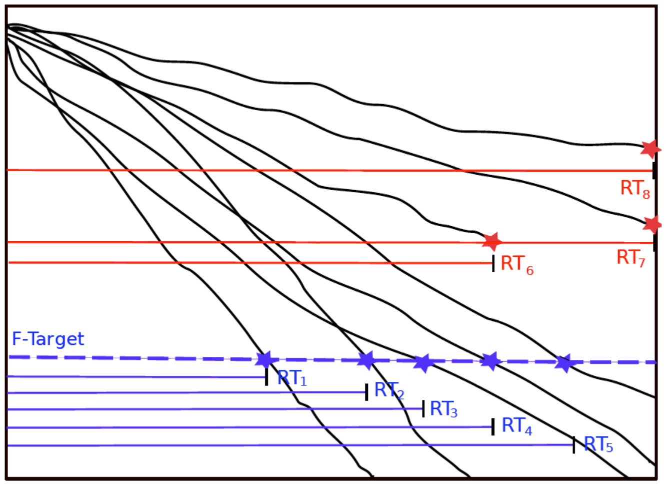
Evaluation of Search Algorithms
Behind the scene

a performance should be
- quantitative on the ratio scale (highest possible)
  - “algorithm A is two *times* better than algorithm B” is a meaningful statement
- can assume a wide range of values

- meaningful (interpretable) with regard to the real world
  - possible to transfer from benchmarking to real world

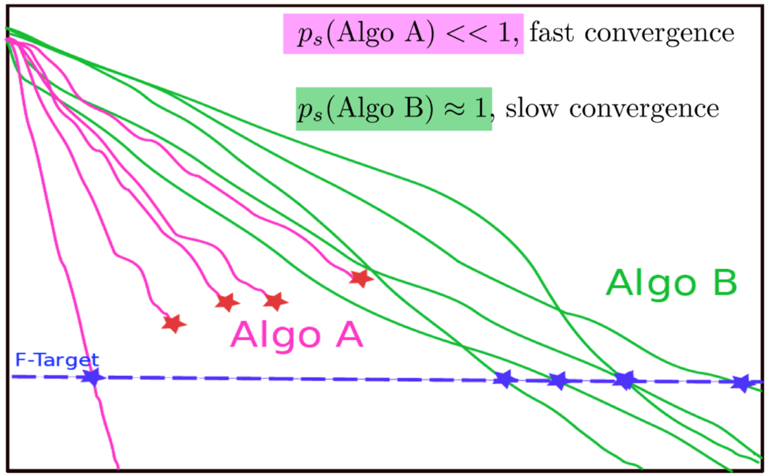
runtime or first hitting time is the prime candidate (we don't have many choices anyway)

Collect Runtime to reach F-target



Which Performance Measure

to compare the two following scenario?



Which Performance Measure

Algo Restart A:

$p_s(\text{Algo Restart A}) = 1$

Algo Restart B:

$p_s(\text{Algo Restart B}) = 1$

Average Runtime (aRT)

$$\text{ERT} = \mathbb{E}[\text{RT}^r] = \frac{1-p_s}{p_s}\mathbb{E}[\text{RT}_{\text{unsuccessful}}] + \mathbb{E}[\text{RT}_{\text{successful}}]$$

aRT is an estimator for ERT

$$\text{aRT} = \frac{\#\text{Evals}}{\#\text{success}}$$

Overview

❶ Problem Statement
- Continuous Black-Box Optimization
- Typical Difficulties

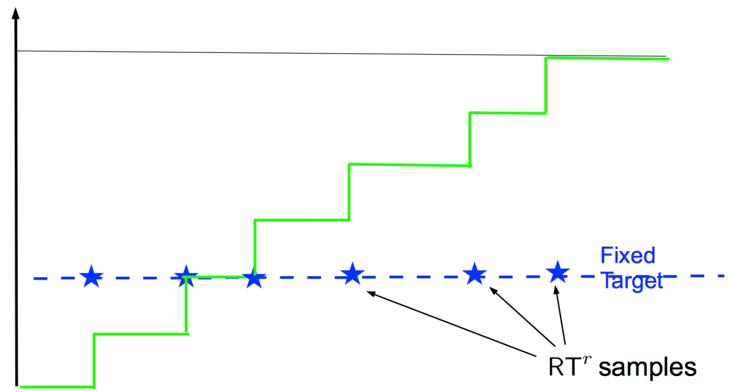
❷ Stochastic Black-Box Algorithms
- General Template
- Invariance
- Comparisons of a few DFOs

❸ Zoom on Evolution Strategies
- Step-size Adaptation
- Covariance Matrix Adaptation

④ Evaluating Black-Box Algorithms
- Displaying results and visualization
- Statistics
- Average Runtime
- Empirical Cumulative Distribution Function (ECDF)

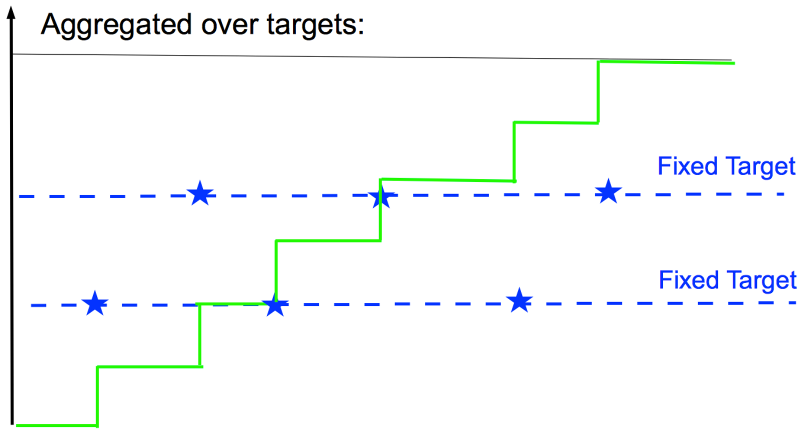
Empirical Cumulative Distribution Function

Single Target

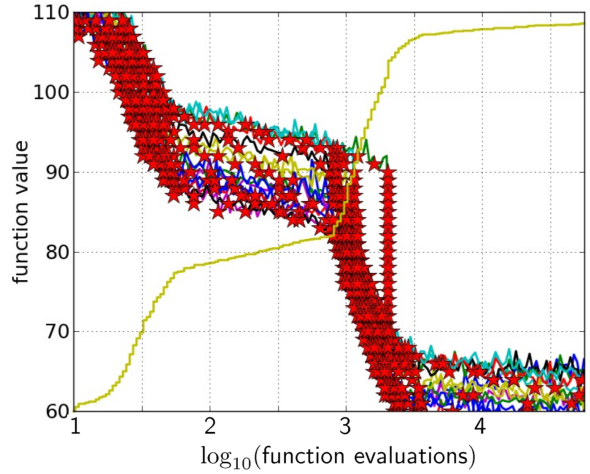


Empirical Cumulative Distribution Function

Several Targets



Empirical Cumulative Distribution Function

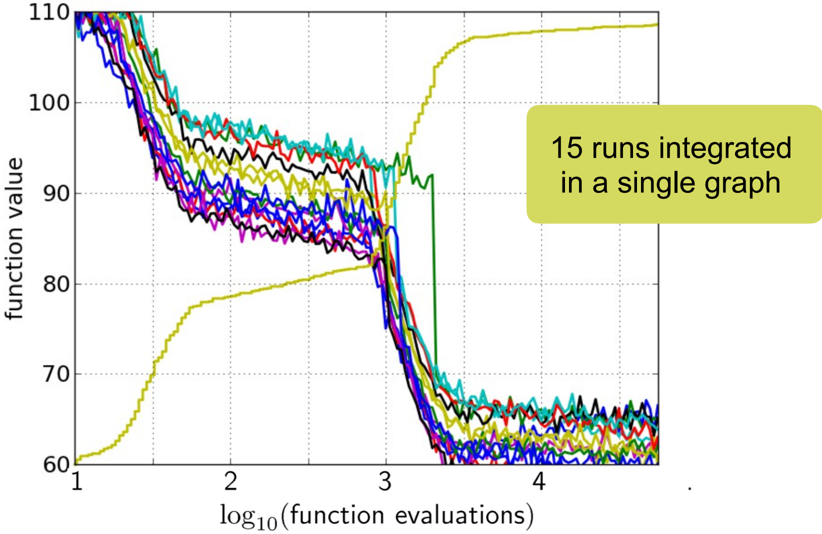


15 runs

50 targets

ECDF with 750 steps

Empirical Cumulative Distribution Function



Empirical Cumulative Distribution Function