Experimental Comparisons of Derivative Free Optimization Algorithms

Anne Auger Nikolaus Hansen J. M. Perez Zerpa
Raymond Ros Marc Schoenauer

TAO Project-Team, INRIA Saclay – Île-de-France, and
Microsoft-INRIA Joint Centre, Orsay, FRANCE
http://tao.lri.fr
First.Last@inria.fr

SEA'09, 4 juin 2009

Problem Statement

Continuous Domain Search/Optimization

The problem

- Minimize a objective function (*fitness* function, *loss* function) in continuous domain

$$f : \mathcal{S} \subseteq \mathbb{R}^n \to \mathbb{R},$$

- in the Black Box scenario (direct search)

Hypotheses

- domain specific knowledge only used within the black box
- gradients are not available

Problem Statement

Continuous Domain Search/Optimization

The problem

- Minimize a objective function (*fitness* function, *loss* function) in continuous domain

$$f : \mathcal{S} \subseteq \mathbb{R}^n \rightarrow \mathbb{R},$$

- in the Black Box scenario (direct search)

Typical Examples

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

The practitionner's point of view

Issues

- How to choose the best algorithm?
  - For a given objective function — set of functions
  - Without theoretical support
- Empirical comparisons on extensive test suites
  - what performance measures?
  - what test functions? — representative of real-world

Some proposals

- Expected Running Time + Empirical Cumulative Distributions
- An artificial testbed, with controlled typical difficulties
- A (partial) case study, involving 2 deterministic and 3 bio-inspired algorithms
- in back-box scenario — without specific intensive parameter tuning

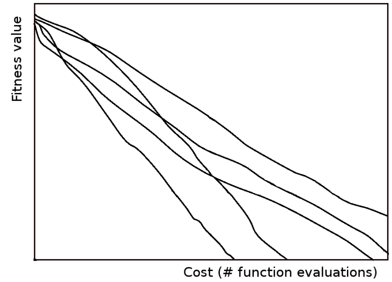
1. Performance Measures and Experimental Comparisons
   - How to empirically compare algorithms?
2. Problem difficulties and algorithm invariances
3. Derivative-Free Optimization Algorithms
4. Experiments and Results
5. Conclusion

Context

Optimization goal(s)

- Find best possible objective value $\neq$ get close to the global optimum
- At minimal cost $\approx$ number of function evaluations

Available data



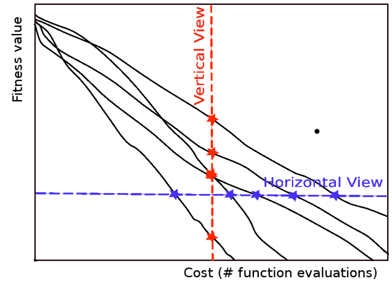
- Objective value vs Computational cost — Minimization
- What to record/analyze/summarize?

Context

Optimization goal(s)

- Find best possible objective value $\neq$ get close to the global optimum
- At minimal cost $\approx$ number of function evaluations

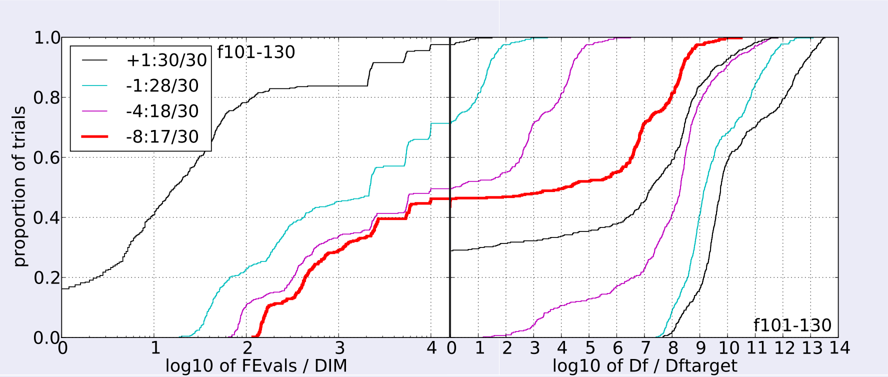
Two points of view



- Vertical: Value reached for a given effort
- Horizontal: Effort required to reach a given objective value

Both Views: Empirical Cumulative Distributions Fns

Horizontal | **Vertical**



- . . . of normalized running times, up to an upper bound ($10^4 n$)
- for different target values 10^{-8}, 10^{-4}, 10^{-1}, 10^1
- . . . of precision, w.r.t. given target value (10^{-8})
- for different max. running times n, $10n$, $100n$, $1000n$

Discussion

Vertical vs horizontal

- Vertical: Value reached for a given effort
  - Fixed budget scenario
  - Qualitative comparisons Algo. A reaches better value than Algo. B
- Horizontal: Effort required to reach a given objective value
  - Baseline requirement e.g. beat the opponent!
  - Absolute comparisons: Algo. A is X times faster than Algo. B
  - Monotonous-invariant criterion

Statistics

- Difficult to summarize multiple viewpoints into a single measure
- . . . and to find a sound estimator for it

compute its variance, perform statistical tests, . . .

Performance measures

ECDFs

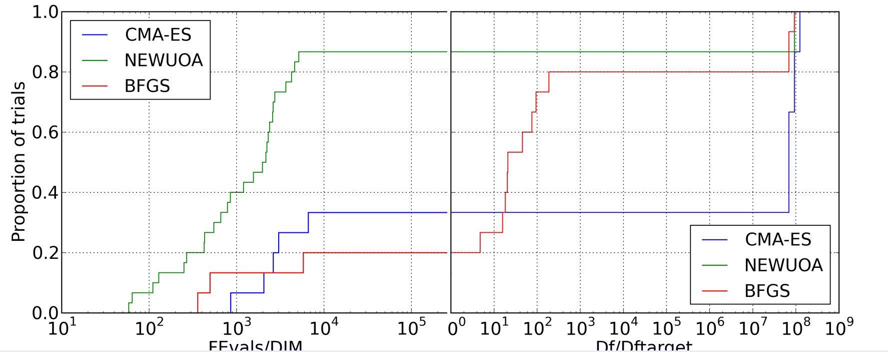
- Require arbitrary
  - maximal target precision,
  - maximal run length
- Can be used for sets of benchmark functions previous slide
- Need to be sub-sampled for comparisons

Horizontal performance measures

- Fix a *target objective value*,
- compute Expected Running Times Distribution,
- measure average effort to success

from Empirical Cumulative Distribution Functions

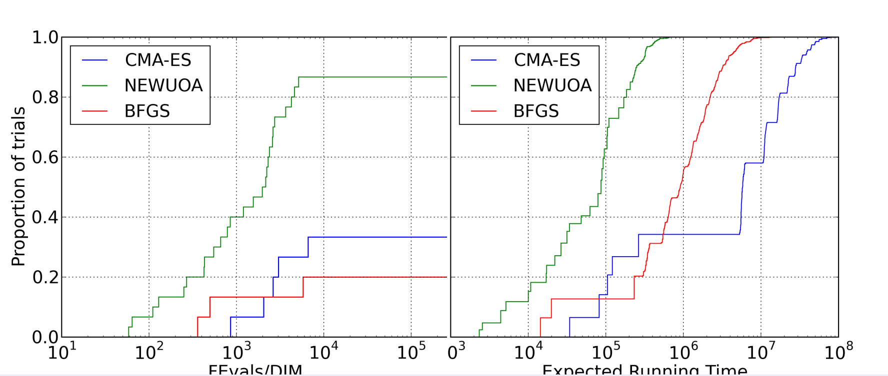
Horizontal | **Vertical**



from Empirical Cumulative Distribution Functions

to Expected Running Time

Runtime ECDF | Estimated Runtime Distribution



using *resampling* technique

Expected Running Time

Experiments and notations

- Fixed number of runs
- Arbitrary target objective value f_{target}
- Arbitrary bound on # evaluations
- p_{succ}: proportion of successful runs that reached f_{target}
- $\widehat{RT}_{succ}$ (resp. $\widehat{RT}_{fail}$): empirical average number of evaluations of successful (resp. unsuccessful) runs

Expected Running Time Measures

$$SP1(f_{target}) = \frac{\widehat{RT}_{succ}}{p_{succ}} \qquad SP2(f_{target}) = \frac{p_{succ}\widehat{RT}_{succ} + (1 - p_{succ})\widehat{RT}_{fail}}{p_{succ}}$$

Expected Running Time (2)

Discussion

- Both measures
  - reflect some average effort to reach f_{target}
  - are equivalent in case of 100% success
  - are unreliable estimators in case of small p_{succ}
  - can be used to easily compare algorithms on sets of functions
    by normalizing w.r.t. best algorithm on each function
- SP1 insensitive to the running length of unsuccessful runs
- SP2 very sensitive to the stopping criterion and the restart strategy, that are part of the algorithm fine tuning ...

History

- CEC'05 Challenge on Continuous Optimization used SP1
- GECCO'09 Workshop on Black-Box Optimization Benchmarking uses SP2

1. Performance Measures and Experimental Comparisons

2. Problem difficulties and algorithm invariances
   - What makes a continuous optimization problem hard?

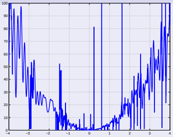
3. Derivative-Free Optimization Algorithms

4. Experiments and Results

5. Conclusion

Problem Difficulties and Algorithm Invariances

What makes a problem hard?



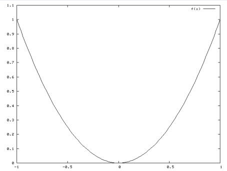
- Non-convexity
  invalidates most of deterministic theory
  - Ruggedness
    non-smooth, discontinuous, noisy
  - Multimodality — presence of local optima
- Dimensionality — line search is 'trivial'
  The magnifiscence of high dimensionality . . .
- Ill-conditioning — Very different scalings along different directions
- Non-separability — Correlated variables

The benefits of invariance

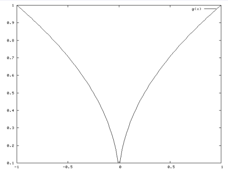
- Some difficulties become harmless
- More robust parameter setting

Ruggedness and Monotonous Invariance

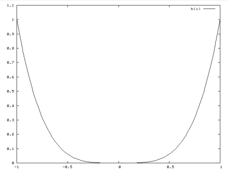
Monotonous transformations of the objective function



$$\sum x_i^2$$



$$\sqrt{\sqrt{\sum x_i^2}}$$



$$\left(\sum x_i^2\right)^2$$

Monotonous Invariance

- Invariance w.r.t. monotonous transformations
- A guarantee against ill-scaled objective functions
- Comparison-based algorithms are monotonous-invariant

Multimodality

Presence of multiple local optima

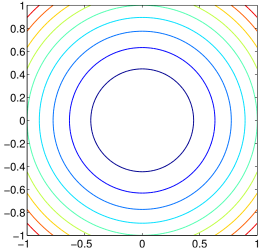
Restart strategies

- For local optimizers, starting point is crucial on multimodal functions
- Multiple restarts are mandatory
  - from uniformly distributed points — global restart
  - from the final point of some previous run after some parameter reset — local restart
- Also efficient with any optimization algorithm

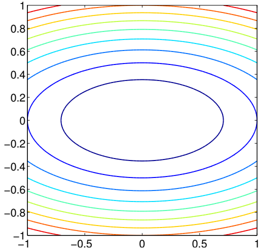
Ill-Conditionning

- The Condition Number (CN) of a positive-definite matrix $\boldsymbol{H}$ is the ratio of its largest and smallest eigenvalues
- If f is quadratic, $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$), the CN of f is that of its Hessian $\boldsymbol{H}$
- More generally, the CN of f is that of its Hessian wherever it is defined.

Graphically, ill-conditioned means "squeezed" lines of equal function value



Increased $\longrightarrow$ condition number



Issue: The gradient does not point toward the minimum ...

Separability

Definition (Separable Problem)

A function f is separable if

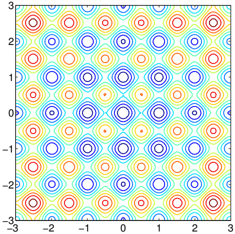
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

solve n independent 1D optimization problems

Example: Additively decomposable functions

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

e.g. Rastrigin function

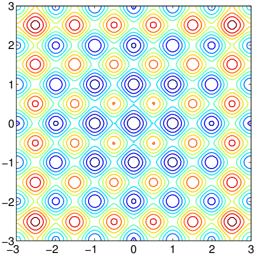


Designing Non-Separable Problems

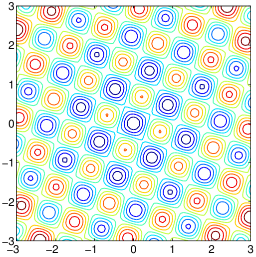
Rotating the coordinate system

- $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ separable
- $f : \boldsymbol{x} \mapsto f(\boldsymbol{R}\boldsymbol{x})$ non-separable $\qquad$ $\boldsymbol{R}$ rotation matrix

Hansen, Ostermeier, & Gawelczyk, 95; Salomon, 96



$\xrightarrow{\boldsymbol{R}}$



Optimization techniques

Numerical methods

- Applied Mathematicians — Long history
- Classical methods based on First Principles — gradient-based
  require regularity — numerical gradient amenable to numerical pitfalls
- Recent DFO methods — quadratic interpolation of the objective function

Bio-inspired algorithms

- Computer Scientists — mostly from AI field
- Many recent (and trendy) methods

but

- Computationally heavy
- No convergence proof — almost . . .

BFGS

Broyden, Fletcher, Goldfarb, & Shanno, 1970

Gradient-based methods

$$\begin{cases} x_{t+1} = x_t - \rho_t d_t \\ \rho_t = Argmin_\rho\{\mathcal{F}(x_t - \rho d_t)\} \end{cases}$$ Line search

Choice of d_t, the descent direction?

BFGS: a Quasi-Newton method

- Maintain an approximation $\hat{H}_t$ of the Hessian of f
- Solve for d_t

$$\hat{H}_t d_t = \nabla f(x_t)$$

- Compute x_{t+1} and update $\hat{H}_t \rightarrow \hat{H}_{t+1}$
- Converges if quadratic approximation of $\mathcal{F}$ holds

around the optimum

- Reliable and robust

on quadratic functions!

BFGS: A priori Discussion

Properties

- Gradient-based algorithms are
  - local optimizers
  - not monotonous invariant
  - independent of the coordinate system
- But numerical gradient is not rotational invariant

and can suffer from ill-conditionning

Implementation

- Matlab built-in `fminunc` — widely blindly used
- using numerical gradient
- Function improvement threshold set to 10^{-25}
- Multiple restarts mandatory — local or global

NEWUOA

Powell, 2006

A Derivative-Free Optimization Algorithm

- Builds a quadratic interpolation of the objective function
- Maintains a trust region size ρ where interpolation is accurate
- Alternates
  - trust region iterations: one conjugate gradient step on the surrogate model
  - alternative iterations: new trust region and surrogate model

Parameters

- Number of interpolation points from $n+6$ to $\frac{n(n+1)}{2}$ — $2n+1$ recommended
- Initial (ρ_{init}) and final (ρ_{end}) sizes of trust region

NEWUOA: A priori Discussion

Properties

- a global optimizer — (re)start with large trust region size
- not monotonous invariant
- Full model ($\frac{n(n+1)}{2}$ points) is independent of coordinate system
  Most efficient model ($2n+1$) is not

Implementation

- Matthieu Guibert's implementation
  `http://www.inrialpes.fr/bipop/people/guilbert/newuoa/newuoa.html`
- $\rho_{init} = 100$ and $\rho_{end} = 10^{-15}$ — after some preliminary experiments
- Local restarts by resetting ρ to ρ_{init}

Stochastic Search

A unified point of view

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 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Covers

- Deterministic algorithms — including BFGS and NEWUOA
- Evolutionary Algorithms, PSO, DE
  P implicitly defined by the variation operators (crossover/mutation)
- Estimation of Distribution Algorithms

Particle Swarm Optimization (PSO)

Eberhart & Kennedy, 1995

The basic algorithm

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$ be a set of particles

1. Sample new positions

$$x_i^j(t+1) = x_i^j(t) + w \underbrace{\left(x_i^j(t) - x_i^j(t-1)\right)}_{\text{aka velocity}}$$

$$+ \underbrace{c_1 \mathcal{U}_i^j(0,1)(p_i^j - x_i^j(t))}_{\text{approach the "previous" best}} + \underbrace{c_2 \tilde{\mathcal{U}}_i^j(0,1)(g_i^j - x_i^j(t))}_{\text{approach the "global" best}}$$

2. Evaluate $\boldsymbol{x}_1(t+1), \ldots, \boldsymbol{x}_\lambda(t+1)$
3. Update distribution parameters

$$\boldsymbol{p}_i = \boldsymbol{x}_i(t+1) \text{ if } f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{p}_i)$$

$$\boldsymbol{g}_i = \boldsymbol{x}_*(t+1) \text{ where } \mathrm{f}(\boldsymbol{x}_*(\mathrm{t}+1)) = \min\{\mathrm{f}(\boldsymbol{x}_\mathrm{k}(\mathrm{t}+1)), \mathrm{k} \in \text{neighbor}(i)\}$$

PSO: A priori Discussion

Properties

- Comparison-based — Monotonous invariance
- not rotational invariant — sampled distribution is an hyperrectangle

Implementation

- Standard PSO 2006, C code,
  $\neq$ Matlab code! from *PSO Central* at
  `http://www.particleswarm.info/`
- Default settings:
  - inertia weight $w \approx 0.7$,
  - $c_1 = c_2 \approx 1.2$
- Swarm size = $\lambda = 10 + floor(2\sqrt{n})$

Differential Evolution (DE)

Rainer & Storn, 1995

Basic algorithm

- Generate NP individuals uniformly in bounds
- Until *stopping criterion*
  - For each individual x in the population
    - Perturbation using an intra-population difference vector

$$\hat{x} = x^{\alpha} + F(x^{\beta} - x^{\gamma})$$

    - (Uniform) crossover with probability $1 - CR$ — Keep $\hat{x}$ if $CR = 1$

$$y_i = \begin{cases} \hat{x}_i \text{ if } U(0,1) < CR \\ x_i \text{ otherwise} \end{cases}$$

    - Keep best from x and y

DE: A priori Discussion

Properties

- Comparison-based — Monotonous invariance
- Crossover is not rotational invariant — exchange coordinates

Implementation

- C code from DE home page
- No default setting!
- Extensive DOE: — On 10-D ellipsoid function
  - Strategy 2
  - $F = 0.8$,
  - $CR = 1.0$ — Rotational invariance
- Population size: $NP = 10 * n$

The $(\mu, \lambda)-$CMA–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 offspring $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ 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

$$\mathcal{N}(\boldsymbol{m}, \sigma^2 \boldsymbol{C}) \sim \boxed{\boldsymbol{m} + \sigma \mathcal{N}(\boldsymbol{0}, \boldsymbol{C})}$$

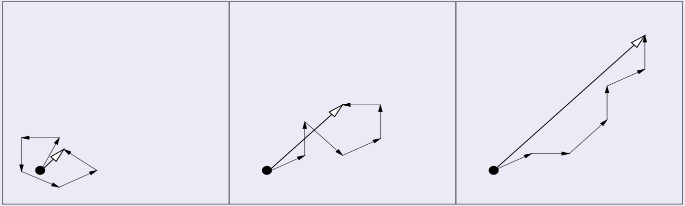
- $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{C}, \sigma\} \in (\mathbb{R}^n \times \mathbb{R}^{n \times n} \times \mathbb{R}_+)$
- $F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$ only depends on the $\mu \le \lambda$ best offspring

Cumulative Step-Size Adaptation (CSA)

$$\begin{aligned} x_i &= m + \sigma z_i \\ m &\leftarrow m + \sigma \langle z \rangle_{\text{sel}} \end{aligned}$$

Measure the length of the *evolution path*

the pathway of the mean vector m in the generation sequence



$\downarrow$ decrease σ $\qquad\qquad$ $\downarrow$ increase σ

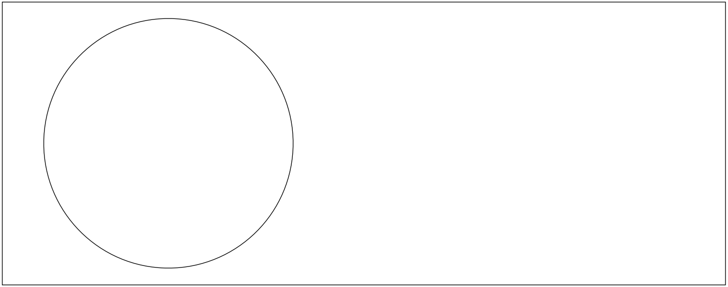
loosely speaking steps are

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

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle z \rangle_{\mathrm{sel}}, \; \langle z \rangle_{\mathrm{sel}} = \sum_{i=1}^{\mu} w_i \, z_{i:\lambda}, \; z_i \sim \mathcal{N}_i(\mathbf{0}, \boldsymbol{C})$$



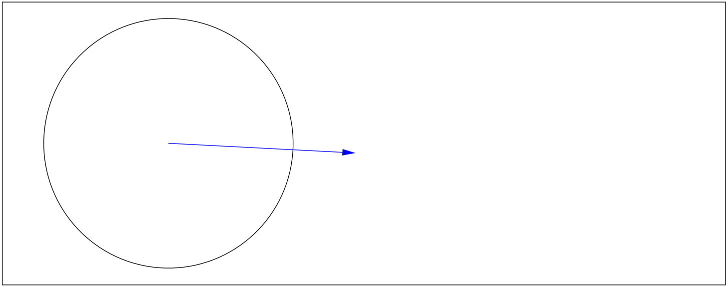
initial distribution, $\boldsymbol{C} = \boldsymbol{I}$

- new distribution: $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle z \rangle_{\mathrm{sel}} \langle z \rangle_{\mathrm{sel}}^{\mathrm{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle z \rangle_{\mathrm{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle \boldsymbol{z} \rangle_{\text{sel}}, \ \langle \boldsymbol{z} \rangle_{\text{sel}} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{z}_{i:\lambda}, \ \boldsymbol{z}_i \sim \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{C})$$



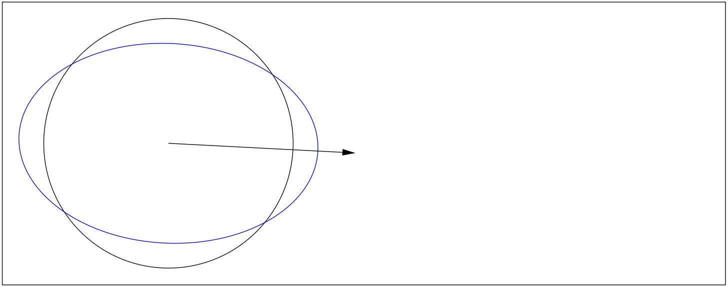
$\langle \boldsymbol{z} \rangle_{\text{sel}}$, movement of the population mean $\boldsymbol{m}$ (disregarding σ)

- new distribution: $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle \boldsymbol{z} \rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle z \rangle_{\text{sel}}, \ \langle z \rangle_{\text{sel}} = \sum_{i=1}^{\mu} w_i \, z_{i:\lambda}, z_i \sim \mathcal{N}_i(\mathbf{0}, \boldsymbol{C})$$



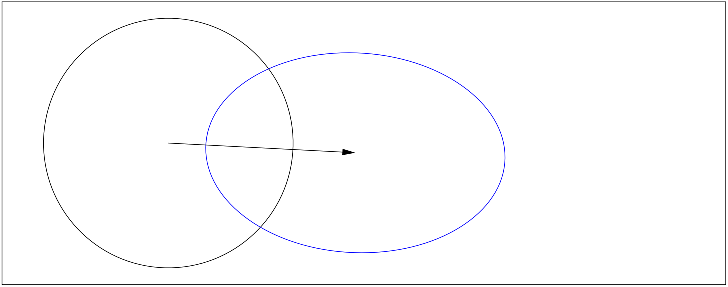
mixture of $\boldsymbol{C}$ and step $\langle z \rangle_{\text{sel}}$, $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle z \rangle_{\text{sel}} \langle z \rangle_{\text{sel}}^{\text{T}}$

- new distribution: $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle z \rangle_{\text{sel}} \langle z \rangle_{\text{sel}}^{\text{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle z \rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma\langle \boldsymbol{z}\rangle_{\text{sel}}, \; \langle \boldsymbol{z}\rangle_{\text{sel}} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{z}_{i:\lambda}, \; \boldsymbol{z}_i \sim \mathcal{N}_i(\boldsymbol{0}, \boldsymbol{C})$$



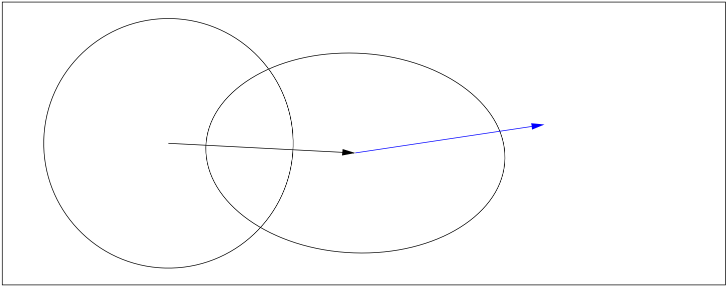
new distribution (disregarding σ)

- new distribution: $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle \boldsymbol{z}\rangle_{\text{sel}} \langle \boldsymbol{z}\rangle_{\text{sel}}^{\mathrm{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle \boldsymbol{z}\rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \langle \boldsymbol{z} \rangle_{\text{sel}}, \ \langle \boldsymbol{z} \rangle_{\text{sel}} = \sum_{i=1}^{\mu} w_i \, \boldsymbol{z}_{i:\lambda}, \boldsymbol{z}_i \sim \mathcal{N}_i(\mathbf{0}, \boldsymbol{C})$$



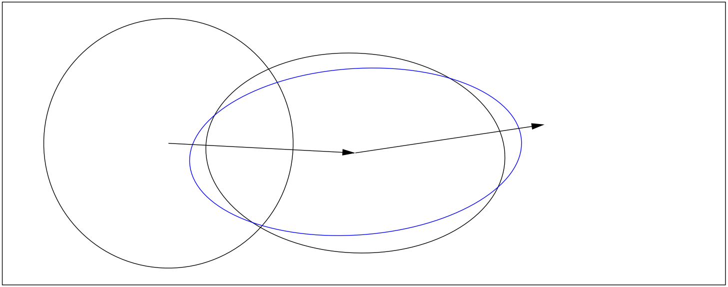
movement of the population mean $\boldsymbol{m}$

- new distribution: $\boldsymbol{C} \leftarrow 0.8 \times \boldsymbol{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle \boldsymbol{z} \rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$m \leftarrow m + \sigma \langle z \rangle_{\text{sel}}, \ \langle z \rangle_{\text{sel}} = \sum_{i=1}^{\mu} w_i \, z_{i:\lambda}, z_i \sim \mathcal{N}_i(\mathbf{0}, C)$$



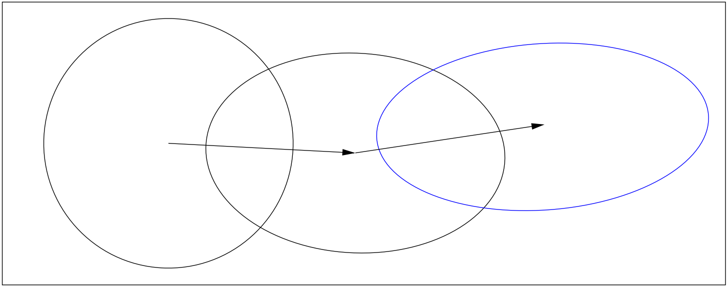
mixture of C and step $\langle z \rangle_{\text{sel}}$, $C \leftarrow 0.8 \times C + 0.2 \times \langle z \rangle_{\text{sel}} \langle z \rangle_{\text{sel}}^{\text{T}}$

- new distribution: $C \leftarrow 0.8 \times C + 0.2 \times \langle z \rangle_{\text{sel}} \langle z \rangle_{\text{sel}}^{\text{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle z \rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

$$m \leftarrow m + \sigma\langle z\rangle_{\mathrm{sel}},\ \langle z\rangle_{\mathrm{sel}} = \sum_{i=1}^{\mu} w_i\, z_{i:\lambda},\ z_i \sim \mathcal{N}_i(\mathbf{0}, C)$$



- new distribution: $C \leftarrow 0.8 \times C + 0.2 \times \langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}}$
- ruling principle: the adaptation increases the probability of successful steps, $\langle z\rangle_{\mathrm{sel}}$, to appear again

CMA-ES: A priori Discussion

Properties

- Comparison-based — Monotonous invariance
- Rotational invariant — Adapts the coordinate system

Implementation

- CMA-ES: Matlab code from author's home page
  http://www.bionik.tu-berlin.de/user/niko/
- Using rank-μ update — faster adaptation Hansen et al., 2003
- Default settings
- Population size: $\lambda = 4 + floor(3 \ln n)$

The parameters

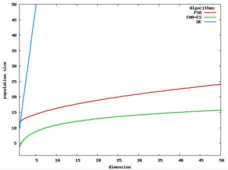
Common Parameters

- Default parameters — except for DE
- Upper bound on run-time = 10^7 evaluations
- $f_{target} = 10^{-9}$
- Domain: $[-20, 80]^d$ — Optimum not at the center
- 21 runs in each case — except BFGS when little success

Population size

Standard values: for $n = 10, 20, 40$

- PSO: $10 + floor(2\sqrt{n})$ — $16, 18, 22$
- CMA-ES: $4 + floor(3 \ln n)$ — $10, 12, 15$
- DE: $10 * n$ — $100, 200, 400$



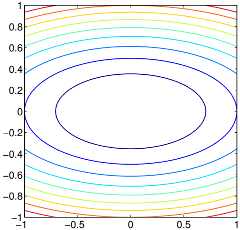
To be increased for multi-modal functions — e.g. Rastrigin

Ellipsoid

- $f_{\text{elli}}(x) = \sum_{i=1}^{n} 10^{\alpha \frac{i-1}{n-1}} x_i^2 = x^T H_{\text{elli}} x$

$$H_{\text{elli}} = \begin{pmatrix} 1 & 0 & \cdots \\ & \ddots & \\ \cdots & 0 & 10^{\alpha} \end{pmatrix}$$

convex, separable

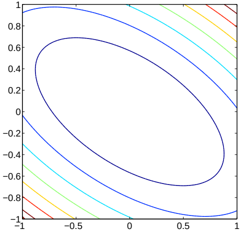


- $f_{\text{elli}}^{\text{rot}}(x) = f_{\text{elli}}(\boldsymbol{R}x) = x^T H_{\text{elli}}^{\text{rot}} x$

$\boldsymbol{R}$ random rotation

$H_{\text{elli}}^{\text{rot}} = \boldsymbol{R}^T H_{\text{elli}} \boldsymbol{R}$

convex, non-separable



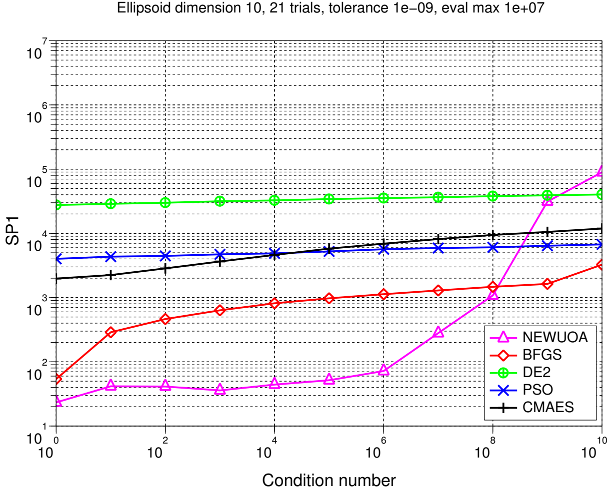
- $\text{cond}(H_{\text{elli}}) = \text{cond}(H_{\text{elli}}^{\text{rot}}) = 10^{\alpha}$

$\alpha = 1, \ldots, 10$

$\alpha = 6 \equiv$ axis ratio of 10^3, typical for real-world problem

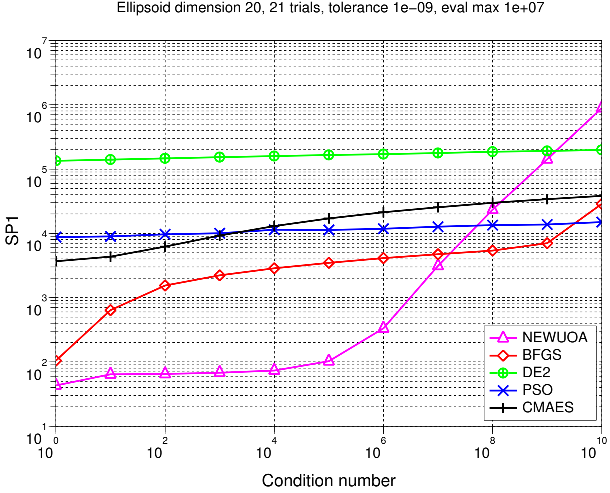
Separable Ellipsoid function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 10



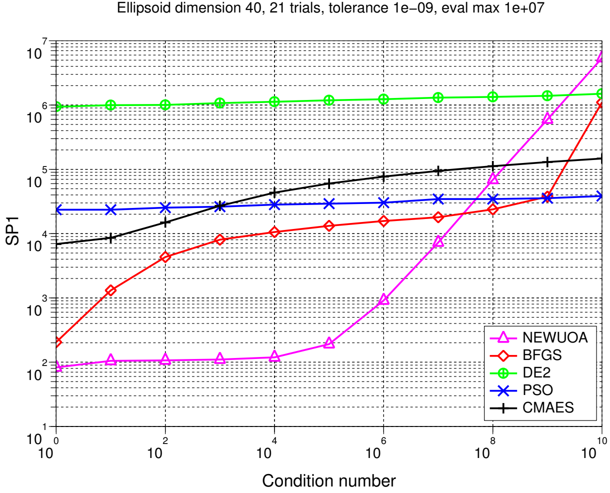
Separable Ellipsoid function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 20



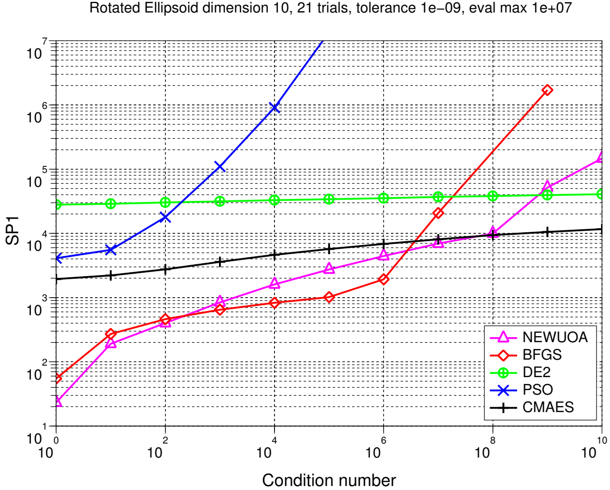
Separable Ellipsoid function

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 40



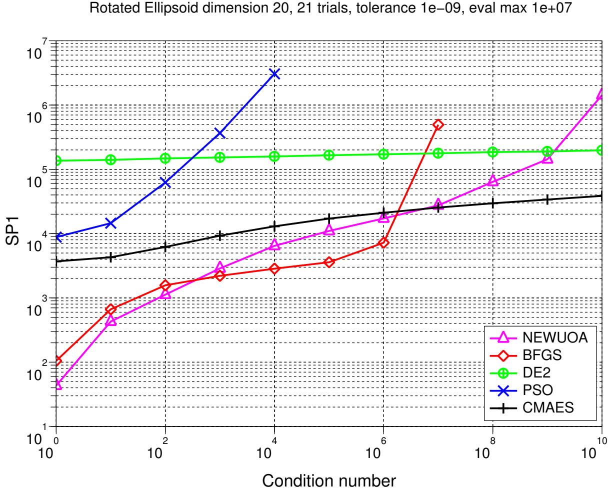
Rotated Ellipsoid function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 10



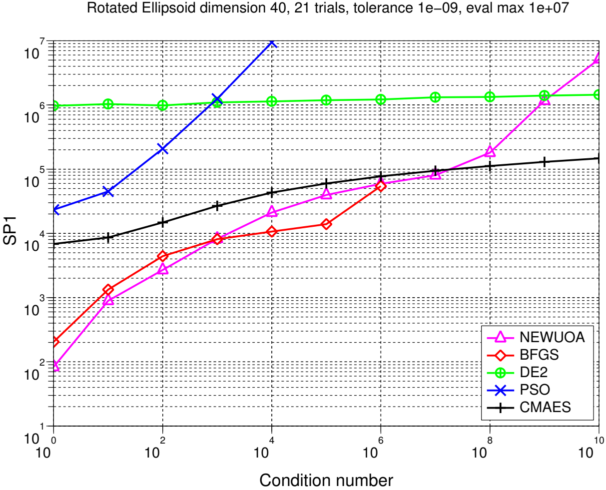
Rotated Ellipsoid function

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 20



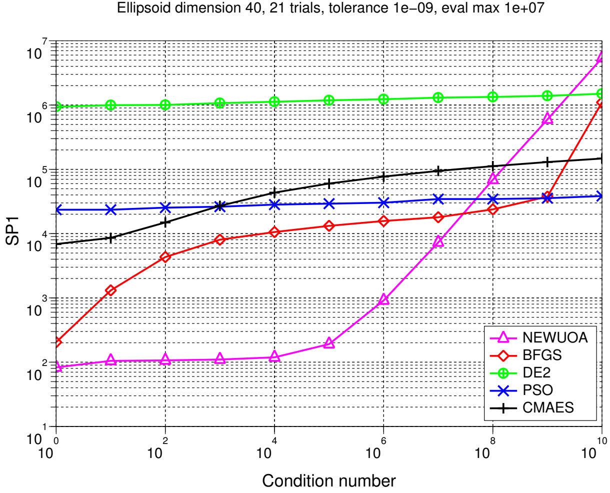
Rotated Ellipsoid function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 40



Separable Ellipsoid function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 40



Ellipsoid: Discussion

Bio-inspired algorithms

- Separable case: PSO and DE insensitive to conditionning
- ...but PSO rapidly fails to solve the rotated version
- ...while CMA-ES and DE ($CR = 1$) are rotation invariant
- DE scales poorly with dimension
  $d^{2.5}$ compared to $d^{1.5}$ for PSO and CMA-ES and BFGS

...vs deterministic

- BFGS fails to solve ill-conditionned cases Matlab “Roundoff error”
- CMA-ES only 7 times slower on pure quadratic functions!
- NEWUOA 70 times better than CMA-ES on separable cases
- performs worse for very high conditionning and rotated cases

Away from "quadraticity"

Monotonous invariance

- Comparison-based algorithms are insensitive to monotonous transformations True for DE, PSO and all ESs
- BFGS and NEWUOA are not theory behind BFGS depends on convexity

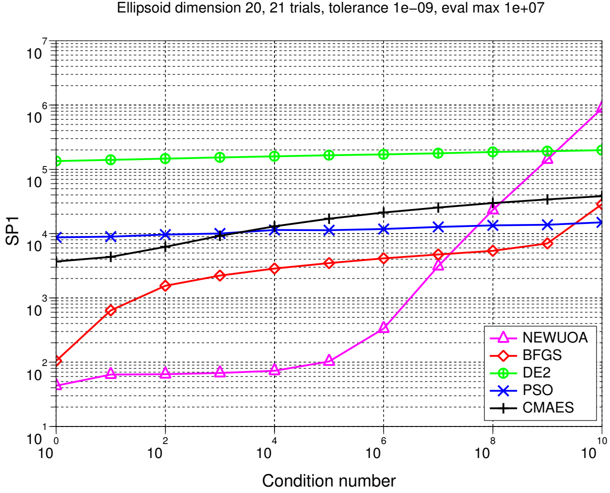
Another test function

- Simple transformation of ellispoid

$$f_{\mathrm{SSE}}(x) = \sqrt{\sqrt{f_{\mathrm{elli}}(x)}}$$

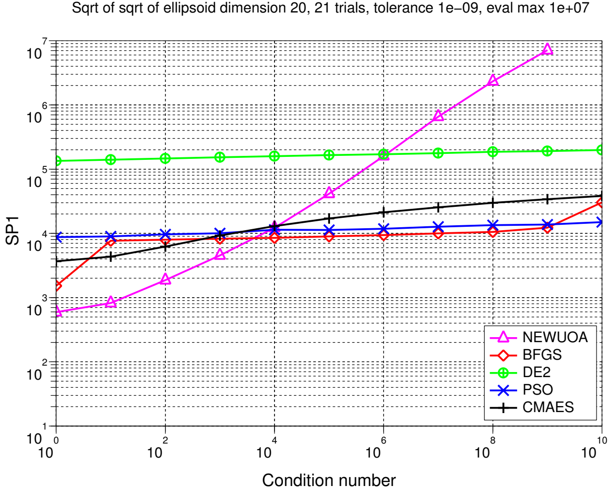
Separable Ellipsoid function

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 20



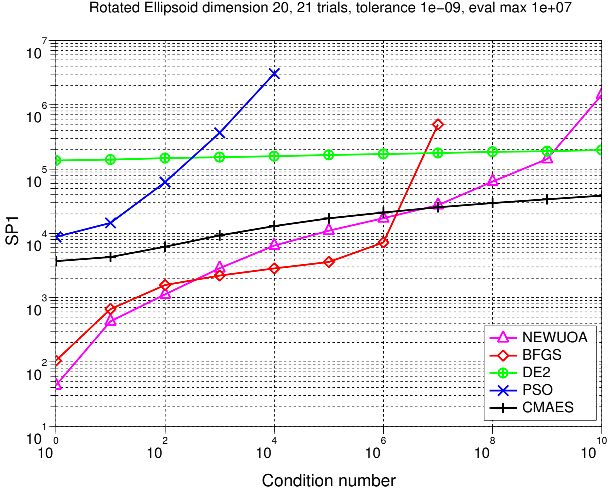
Separable Ellipsoid$^{\frac{1}{4}}$ function

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 20



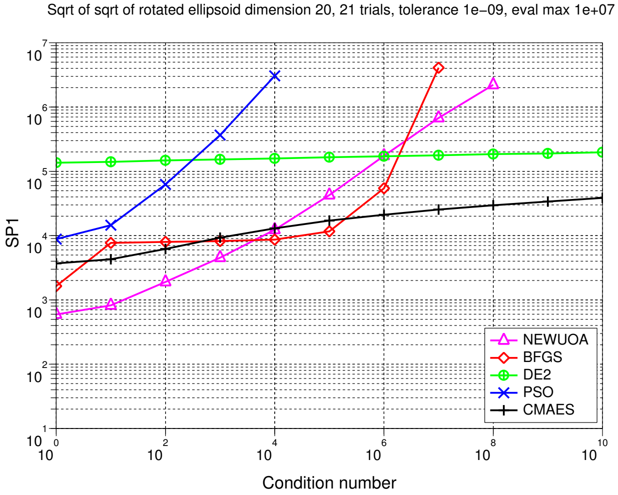
Rotated Ellipsoid function

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 20



Rotated Ellipsoid$^{\frac{1}{4}}$ function

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 20



Non-quadratic: Discussion

Bio-inspired algorithms

- Invariant as expected!

NEWUOA and BFGS

- Worse on $\sqrt{\sqrt{\text{Ellipsoid}}}$ than on Ellipsoid
- Premature numerical convergence for high CN for BFGS . . . fixed by the 'local restart' strategy
- NEWUOA suffers from conjonction of rotation, high CN and non-quadraticity

Rosenbrock function (Banana)

$$f_{\text{rosen}}(x) = \sum_{i=1}^{n-1} \left[(1 - x_i)^2 + \beta(x_{i+1} - x_i^2)^2\right]$$

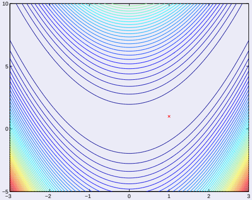
- Non-separable, but ...

  also ran rotated version

- $\beta = 100$, classical Rosenbrock function

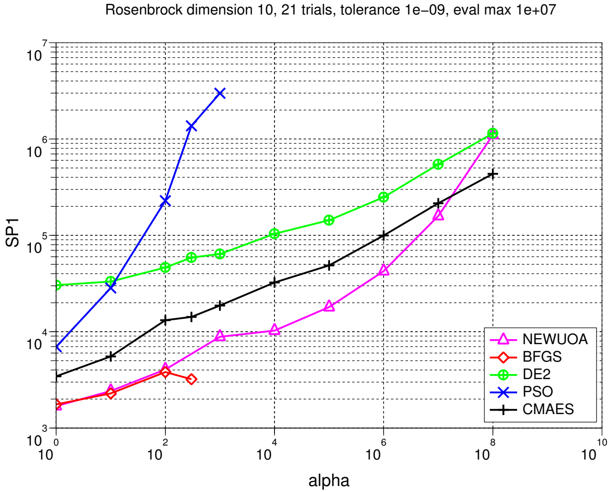
  $\beta = 1, \ldots, 10^8$

- Multi-modal for dimension > 3



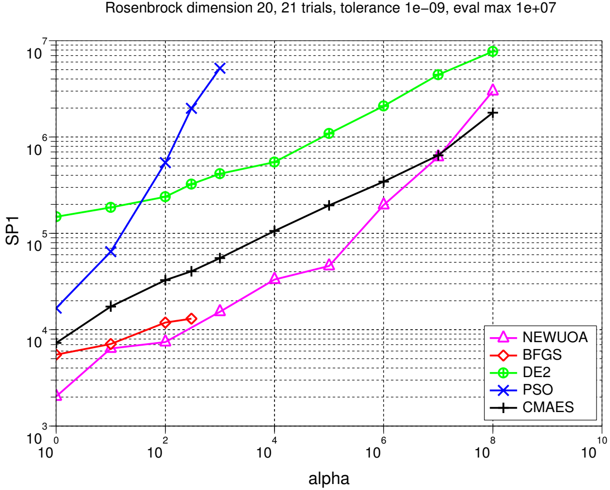
Rosenbrock functions

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 10



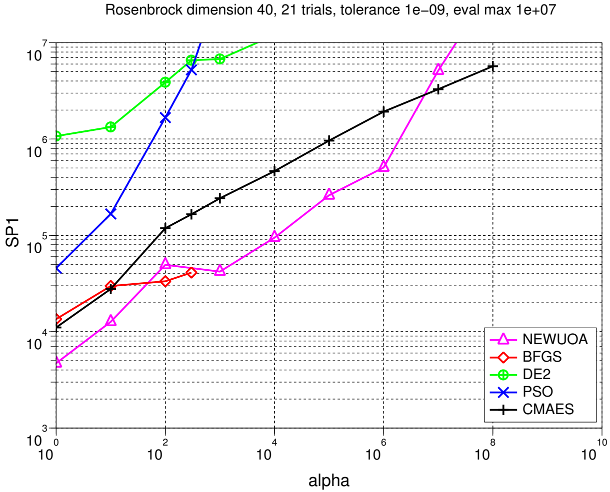
Rosenbrock functions

PSO, DE2, CMA-ES, NEWUOA, and BFGS - Dimension 20



Rosenbrock functions

PSO, **DE2**, **CMA-ES**, **NEWUOA**, and **BFGS** - Dimension 40



Rosenbrock: Discussion

Bio-inspired algorithms

- PSO sensitive to non-separability
- DE still scales badly with dimension

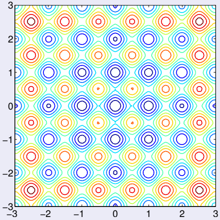
. . . vs BFGS

- Numerical premature convergence on ill-condition problems
- Both local and global restarts improve the results

Rastrigin function

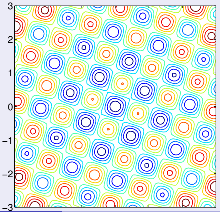
$$f_{\text{rast}}(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\,\cos(2\pi x_i)$$

- separable
- multi-modal



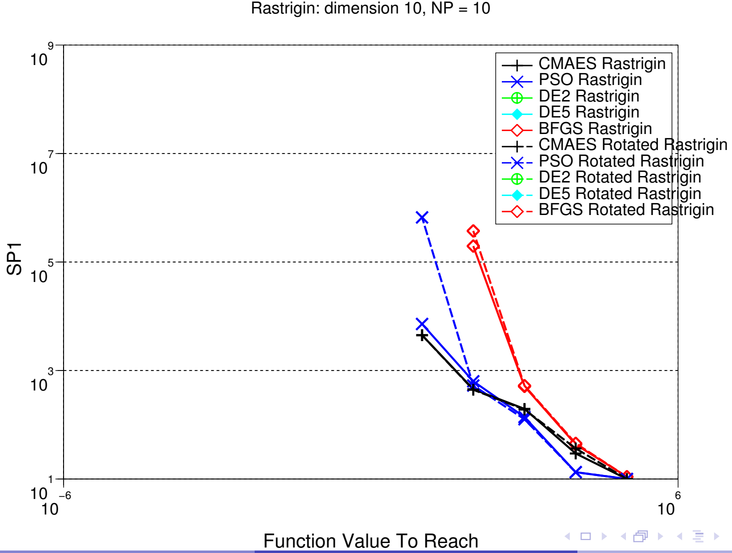
$$f_{\text{rast}}^{\text{rot}}(x) = f_{\text{rast}}(\boldsymbol{R}x)$$

- $\boldsymbol{R}$ random rotation
- non-separable
- multimodal



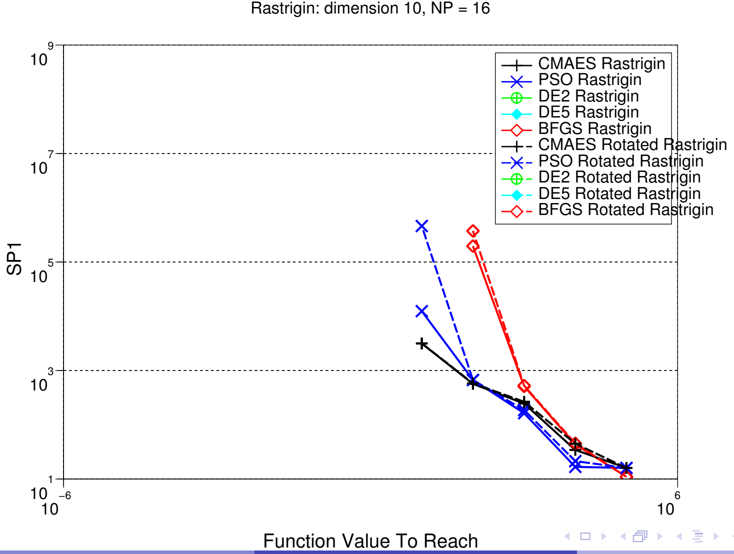
Rastrigin function - SP1 vs objective value

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 10



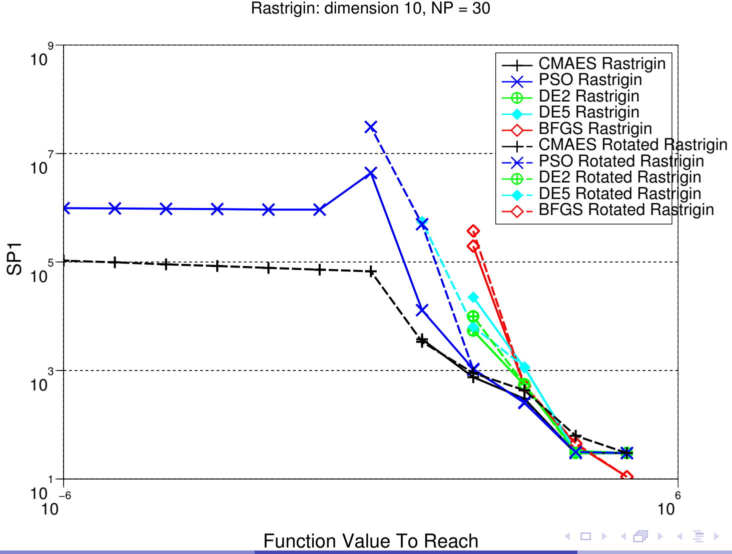
Rastrigin function - SP1 vs objective value

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 16



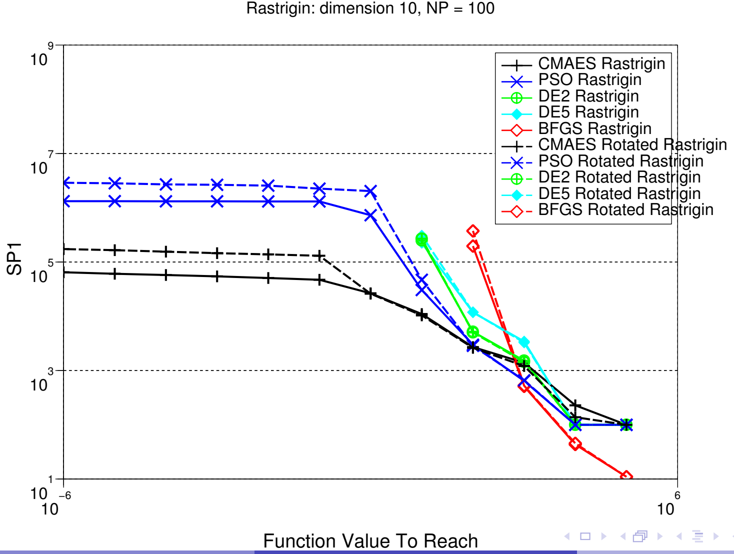
Rastrigin function - SP1 vs objective value

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 30



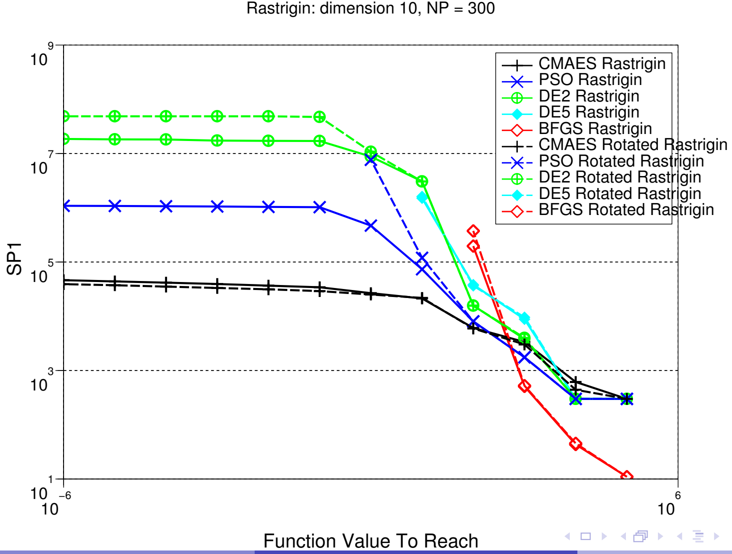
Rastrigin function - SP1 vs objective value

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 100



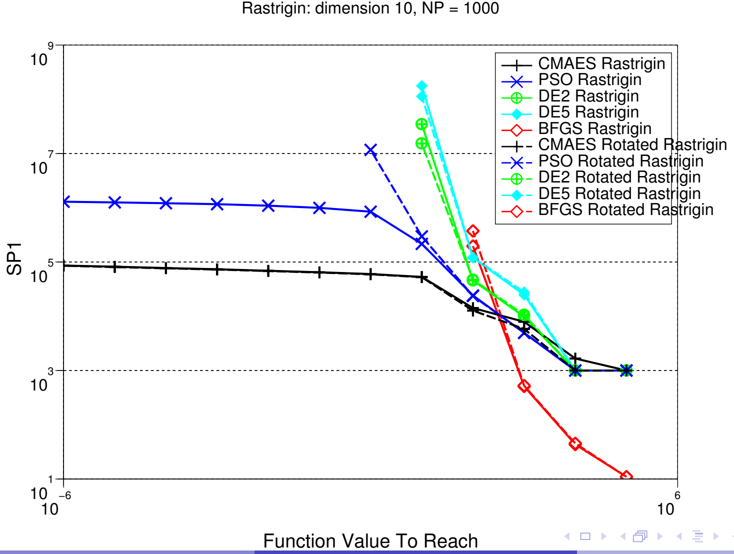
Rastrigin function - SP1 vs objective value

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 300



Rastrigin function - SP1 vs objective value

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 1000

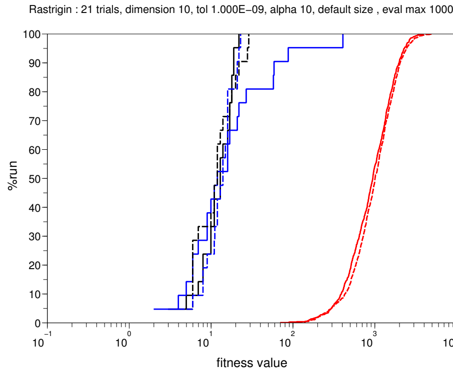


Rastrigin function - Cumulative distributions

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 10

astrigin : 21 trials, dimension 10, tol 1.000E−09, alpha 10, default size , eval max 10000000

evaluation number



% success
vs
eval to reach success threshold ($= 10^{-9}$)

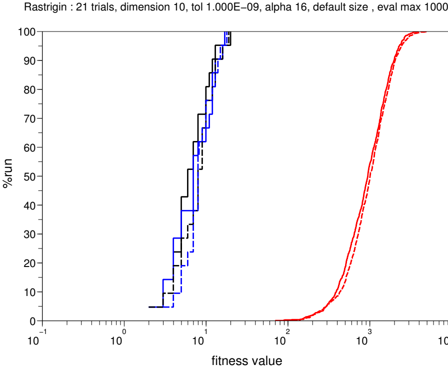
% success
vs
objective value reached before max eval ($= 10^{7}$)

Rastrigin function - Cumulative distributions

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 16

astrigin : 21 trials, dimension 10, tol 1.000E−09, alpha 16, default size , eval max 10000000

evaluation number

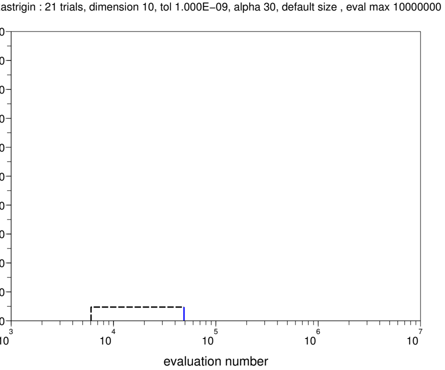


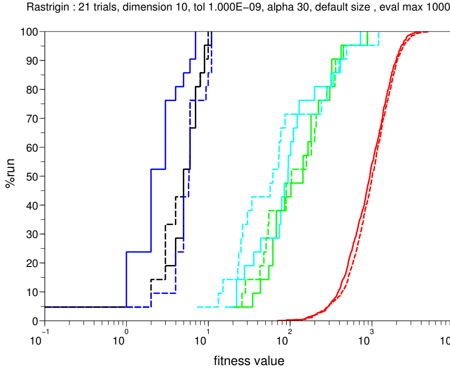
% success
vs
eval to reach success threshold $(= 10^{-9})$

% success
vs
objective value reached before max eval $(= 10^{7})$

Rastrigin function - Cumulative distributions

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 30



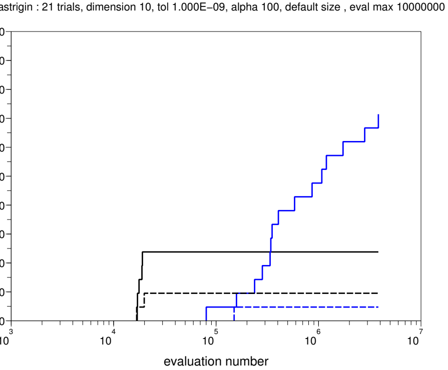


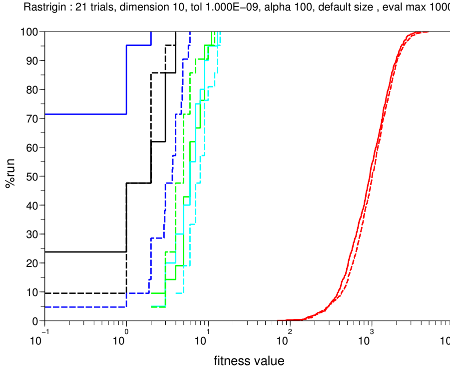
% success
vs
eval to reach success threshold ($= 10^{-9}$)

% success
vs
objective value reached before max eval ($= 10^7$)

Rastrigin function - Cumulative distributions

PSO, DE2, DE5, CMA-ES, and BFGS - PopSize 100



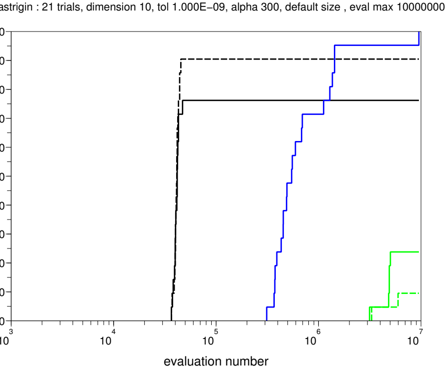


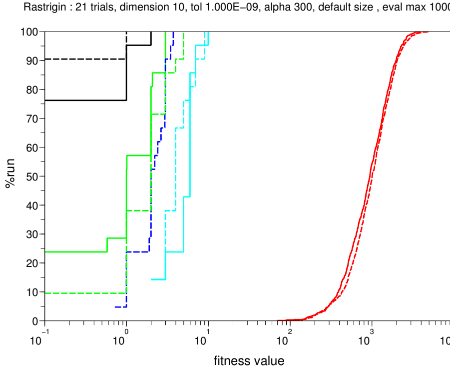
% success
vs
eval to reach success threshold ($= 10^{-9}$)

% success
vs
objective value reached before max eval ($= 10^{7}$)

Rastrigin function - Cumulative distributions

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 300



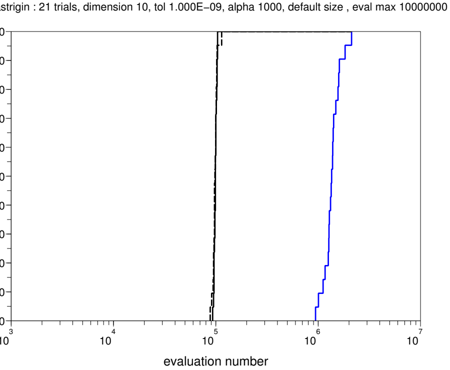


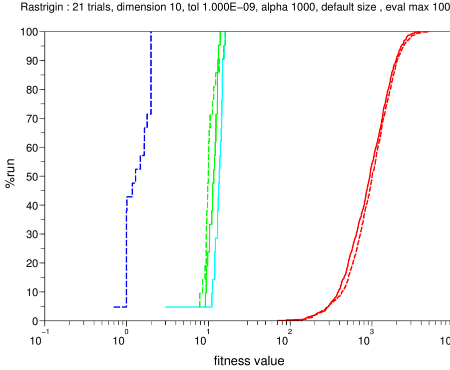
% success
vs
eval to reach success threshold ($= 10^{-9}$)

% success
vs
objective value reached before max eval ($= 10^{7}$)

Rastrigin function - Cumulative distributions

PSO, **DE2**, **DE5**, **CMA-ES**, and **BFGS** - PopSize 1000





% success
vs
eval to reach success threshold ($= 10^{-9}$)

% success
vs
objective value reached before max eval ($= 10^7$)

Rastrigin: Discussion

Bio-inspired algorithms

- Increasing population size improves the results

  Optimal size is algorithm-dependent
- CMA-ES and PSO solve separable case — PSO 100 times slower
- Only CMA-ES solves the rotated Rastrigin reliably

  requires popSize ≥ 300

. . . vs BFGS

- Gets stuck in local optima
- Whatever the restart strategies

  No numerical premature convergence

. . . and NEWUOA?

- solves it in 5 iterations! — identifies the global parabola

Some functions from GECCO'09 BBOB Workshop

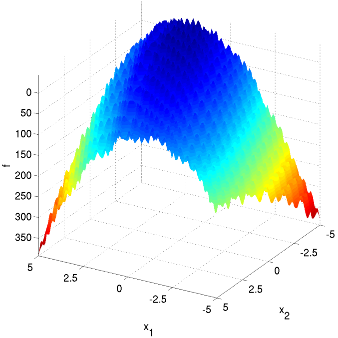
Properties

- No tunable difficulty — e.g. single condition number
- but systematic non-linear transformations
- and asymetrisation of global and local structures

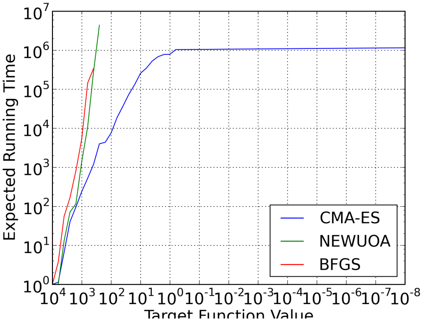
Issue

- Difficult to interpret or generalize
- Except on some families of functions — e.g. with high conditionning

Asymetric Rastrigin



2D plot

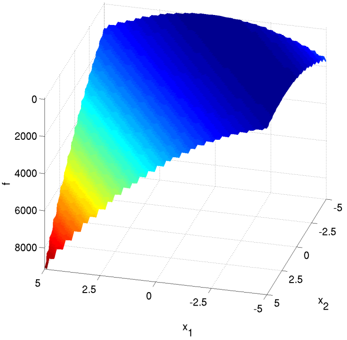


SP2 for $f_{target} = 10^{-8}$

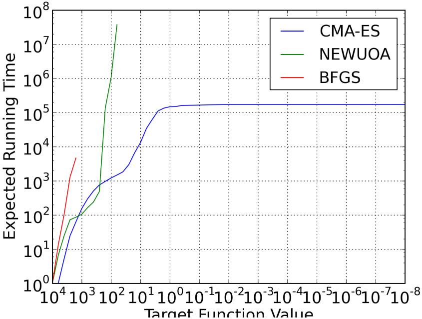
CMA-ES, NEWUOA, and BFGS

Step ellipsoid

Piecewise constant with global ellipsoid structure



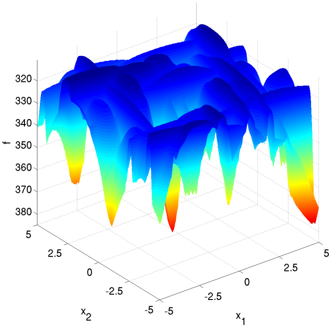
2D plot



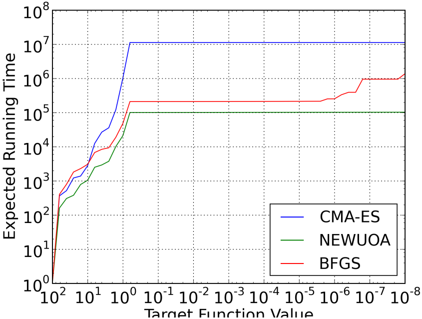
SP2 for $f_{target} = 10^{-8}$
CMA-ES, NEWUOA, and BFGS

Gallagher's Gaussian Peaks

Very weak global structure



2D plot



SP2 for $f_{target} = 10^{-8}$

CMA-ES, NEWUOA, and BFGS

1. Performance Measures and Experimental Comparisons
2. Problem difficulties and algorithm invariances
3. Derivative-Free Optimization Algorithms
4. Experiments and Results
5. **Conclusion**
   - and perspectives

Empirical Comparisons of DFO algorithms

An ill-defined problem

- Trade-off between precision and speed
- Task-dependent

Performance Measures

- Empirical Cumulative Distributions display all information
- Otherwise, need to chose one view-point e.g. horizontal
- → Expected Running Times for easy comparisons

Identified problem difficulties

- allow us to define test-suite for specific comparison purposes
- highlight the benefits of algorithm invariance properties

Bio-inspired Algorithms: The coming of age

Bio-inspired vs Deterministic

- CMA-ES only 7 (70) times slower than BFGS (DFO)

on (quasi-)quadratic functions.

but

- is less hindered by high conditionning,
- is monotonous-transformation invariant,
- is a global search method!

Moreover,

- Theoretical results are catching up
  - Linear convergence for SA-ES with bound on the CV speed — Auger, 05
  - On-going work for CMA-ES

Perspectives

Empirical comparisons

- GECCO'09 BBOB Workshop and Challenge
  - Noise, and/or non-linear asymmetrizing transformations
  - Compare wide set of algorithms — optimally tuned
- Longer term
  - Constrained functions
  - Real-world functions — but which ones ???

COCO, an open platform for COmparison of Continuous Optimizers

Toward automatic algorithm choice

- Need descriptors of problem difficulty — easy to compute
- informative enough to guide algorithmic choice

Inspiration from SAT domain (Hoost et al, 06) and Statistical Physics?