Gaussian processes in evolutionary black-box search

Lukáš Bajer[1,2], Andrej Kudinov[3]

[1]Faculty of Mathematics and Physics, Charles University,
[2]Institute of Computer Science, Czech Academy of Sciences, and
[3]Faculty of Information Technologies, Czech Technical University

Prague, Czech Republic

21. 5. 2015

Contents

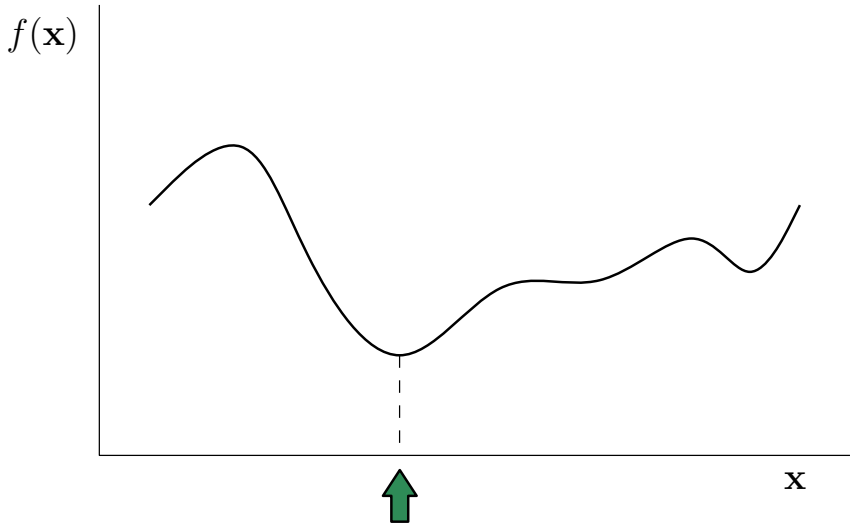
1. Optimization of expensive black-box functions
   - Model-based methods
   - Gaussian Processes
2. MGSO
   - Introduction to MGSO
   - Experimental results on Niching functions
   - Experimental results on BBOB functions
3. Surrogate CMA-ES
   - CMA-ES
   - Surrogate CMA-ES
   - Experimental results

Optimization

- **optimization** (minimization) is finding such $\mathbf{x}^\star \in \mathbb{R}^n$ that

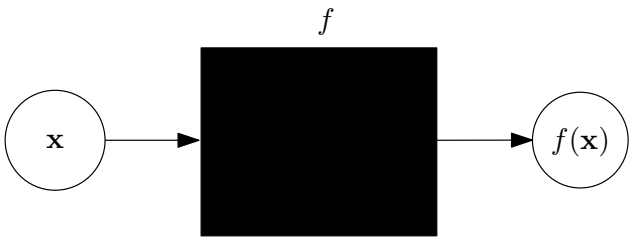
$$f(\mathbf{x}^\star) = \min_{\forall \mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

- "near-optimal" solution is usually sufficient

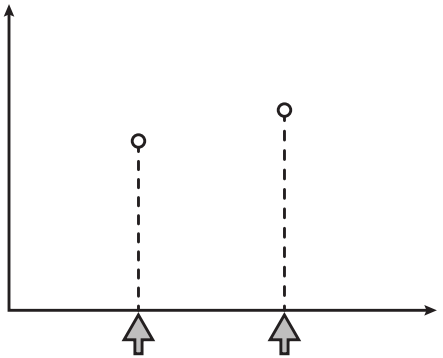


Optimization of **black-box functions**

- **black-box functions**



- only evaluation of the function value, no derivatives or gradients $\rightarrow$ no gradient methods available

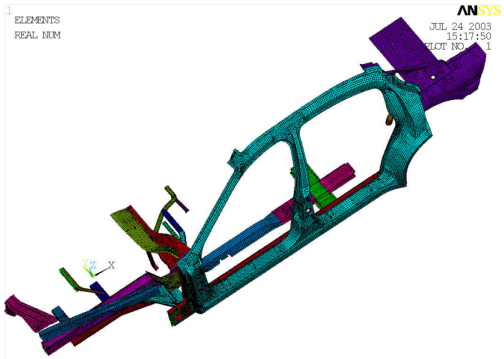


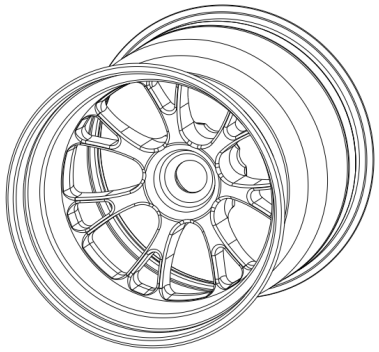
- we consider continuous domain: $\mathbf{x} \in \mathbb{R}^n$

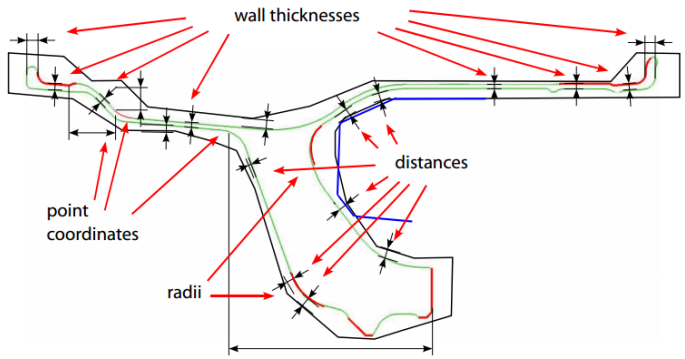
Optimization of **empirical** black-box functions

empirical function:

- assessing the function-value via an experiment (measuring, intensive calculation, evaluating a prototype)
- evaluating such functions are expensive (time and/or money)
- search cost $\sim$ the number of function evaluations

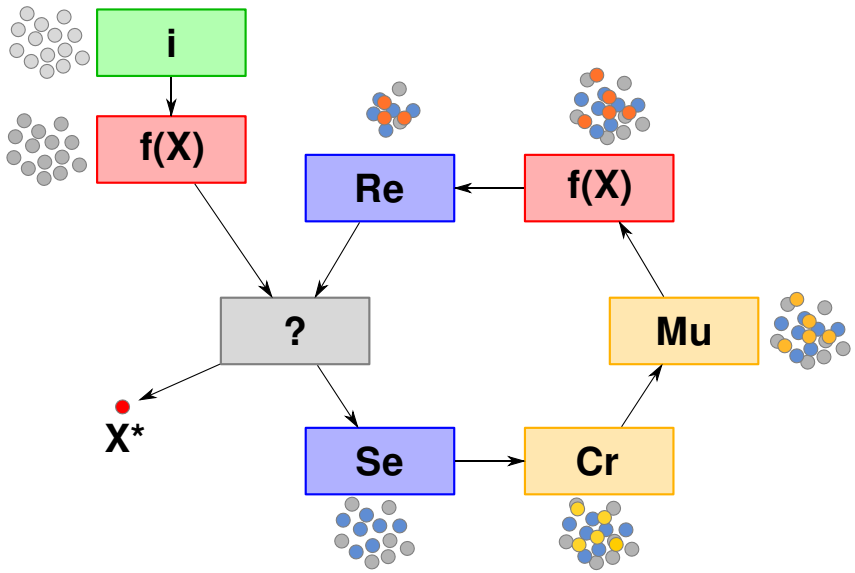






EA's for empirical black-box optimization

- evolutionary algorithms (EA's) used in this study
- EA's often manage to escape from local optima
- but usually use many function evaluations
  (at least in comparison with gradient methods like BFGS)

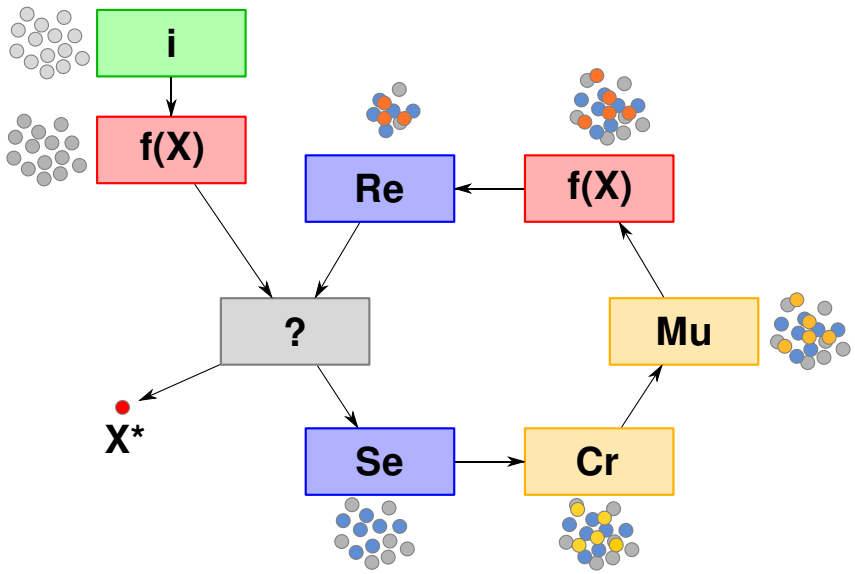


Johann Dréo (CC)

EA's for empirical black-box optimization

what can help with decreasing
the number of function evaluations:

- utilize **already measured values**
  (at least prevent measuring the same thing twice)
- learn **the shape** of the function landscape
  or learn the (global) **gradient** or step direction & size



Model-based methods accelerating the convergence

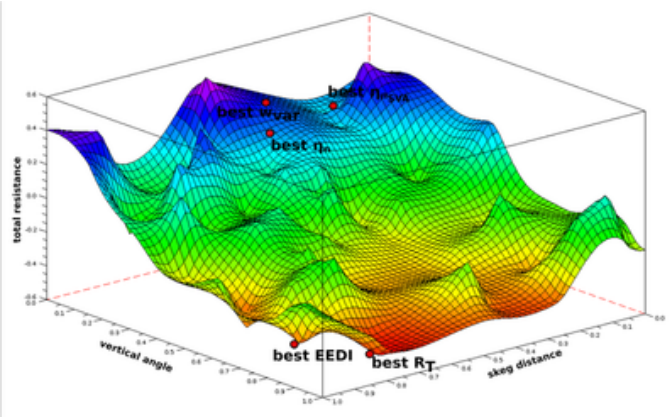
several methods are used in order to **decrease**
the number of objective function **evaluations** needed by EA's

1. Surrogate modelling
2. Estimation of Distribution Algorithms (EDA's)
3. Efficient Global Optimization (EGO)

Surrogate modelling

Surrogate modelling

- technique which builds an **approximating model** of the fitness function landscape
- the model provides a cheap and fast, but also inaccurate replacement of the fitness function for part of the population
- inaccurate approximating model can **deceive** the optimizer



from the EUMC presentation "Viscous optimization of bulkers and tankers" (Mattia Brenner, June 17, 2010)

Estimation of Distribution Algorithms (EDA)

Estimation of Distribution Algorithms

- the model represents a distribution of solutions in the input space
- new candidate solutions are generated via sampling the model
- better solutions are selected for being represented by the model in the next generation

Stochastic search of Evolutionary algorithms

Stochastic black box search

initilize distribution parameters θ
set population size $\lambda \in \mathbb{N}$
while not terminate

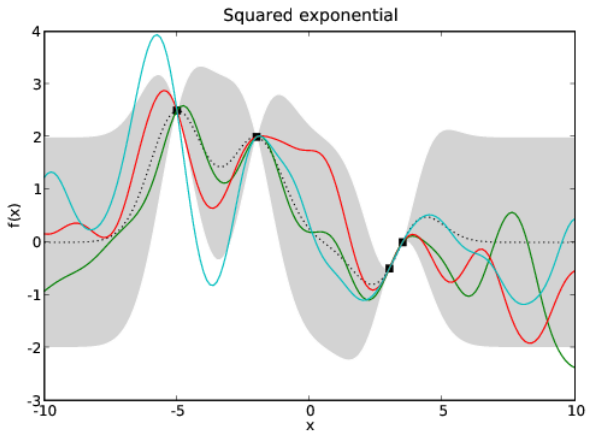
1. sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. evaluate $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ on f
3. update parameters θ

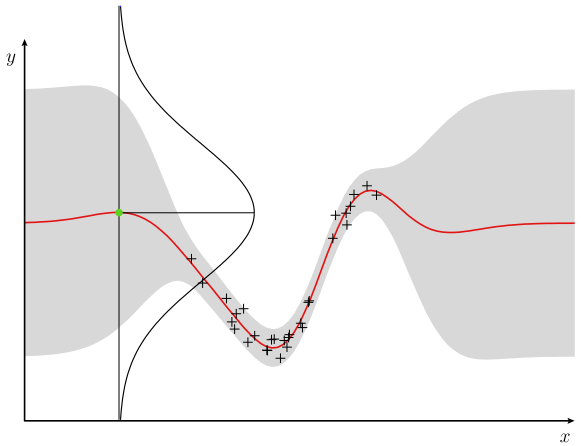
(A. Auger, Tutorial CMA-ES, GECCO 2013)

- schema of most of the evolutionary algorithms and EDA algorithms

Gaussian Process

GP is a stochastic approximation method based on Gaussian distributions

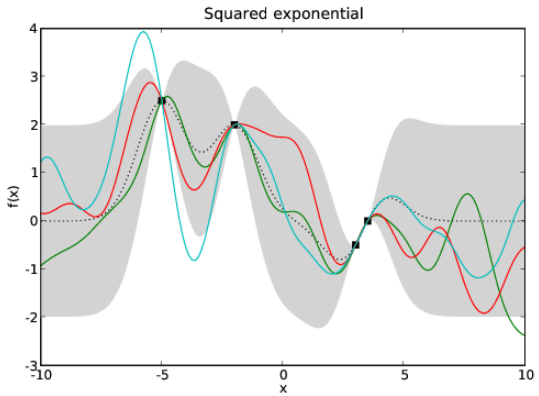




from *infpy* documentation

GP can express **uncertainty** of the prediction in a new point $\mathbf{x}$: it gives a probability distribution of the output value

Gaussian Process



from *infpy* documentation

- given a set of N training points $\mathbf{X}_N = (\mathbf{x}_1 \dots \mathbf{x}_N)^\top$, $\mathbf{x}_i \in \mathbb{R}^d$, and measured values $\mathbf{y}_N = (y_1, \dots, y_N)^\top$ of a function f being approximated

$$y_i = f(\mathbf{x}_i), \quad i = 1, \dots, N$$

GP considers vector of these function values as a sample from N-variate Gaussian distribution

$$\mathbf{y}_N \sim \mathrm{N}(\mathbf{0}, \mathbf{C}_N)$$

Gaussian Process

$\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_N)^\top,\ \mathbf{x}_i \in \mathbb{R}^d$ N training data points

$\mathbf{y} = (y_1, \dots, y_N)^\top,\ y_i = f(\mathbf{x}_i)$ measured function values

$\mathbf{y} \in \mathbb{R}^N$ considered to be a realisation of a N-dimensional Gaussian distribution with a covariance matrix $\mathbf{C}_N$ and zero mean

$$\mathbf{y} \sim \mathrm{N}(\mathbf{0}, \mathbf{C}_N)$$

Covariance $\mathbf{C}_N$ is determined by

- covariance function $cov(\mathbf{x}_i, \mathbf{x}_j)$ and its hyperparameters
- training data points $\mathbf{X}_N$

forming the density of the Gaussian $p(\mathbf{y}|\mathbf{X}_N)$

Gaussian Process covariance

covariance $\mathbf{C}_N$ is given by

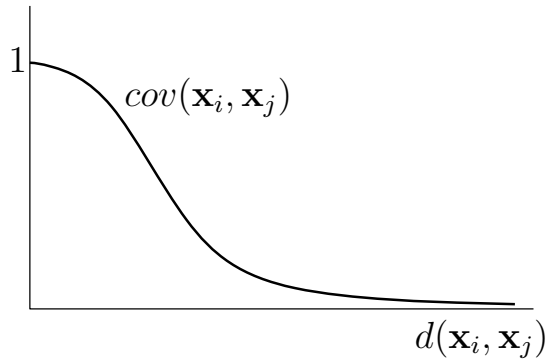
$$\mathbf{C}_N = \mathbf{K} + \sigma^2 \mathbf{I}$$

where $\mathbf{K}$ is a matrix of **covariance function** values and σ^2 is the signal noise.

Covariance functions are defined on pairs from the input space

$$(\mathbf{K})_{ij} = cov(\mathbf{x}_i, \mathbf{x}_j), \quad \mathbf{x}_{i,j} \in \mathbb{R}^d$$

expressing the degree of correlations between two points' values; typically decreasing functions on two points distance



Gaussian Process covariance

The most frequent covariance function is *squared-exponential*

$$(\mathbf{K})_{ij} = cov^{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \theta \exp\left(\frac{-1}{2\ell^2}(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j)\right)$$

with the parameters (usually fitted by MLE)

- θ – signal variance (scales the correlation)
- ℓ – characteristic length scale

Gaussian Process covariance

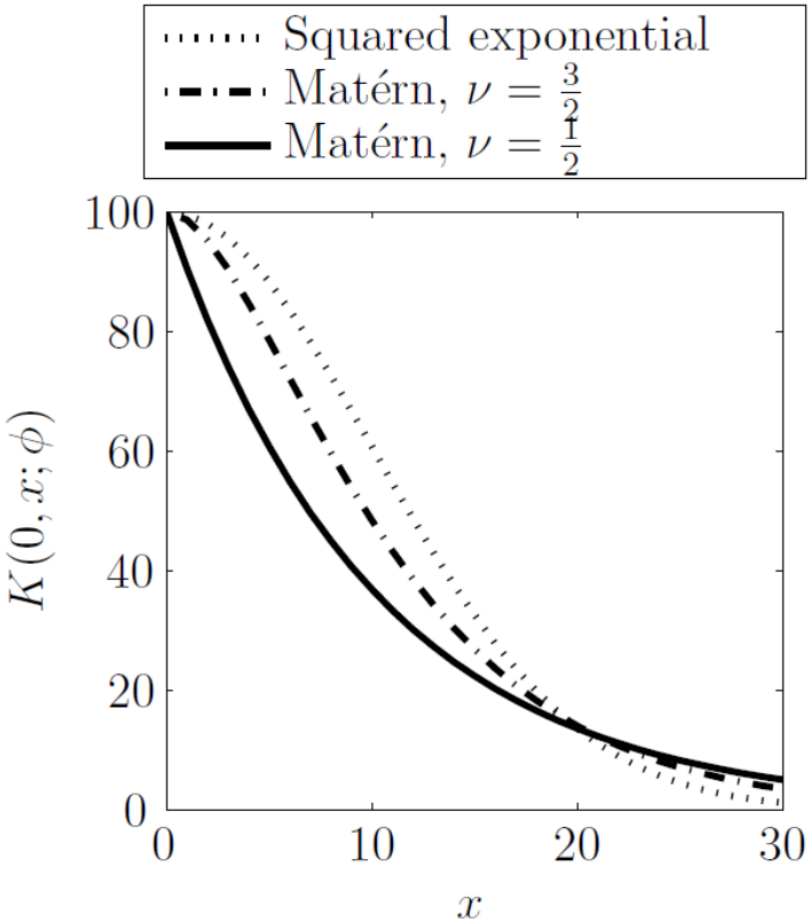
Another usual option is *Matérn covariance*, which is for $r = (\mathbf{x}_i - \mathbf{x}_j)$

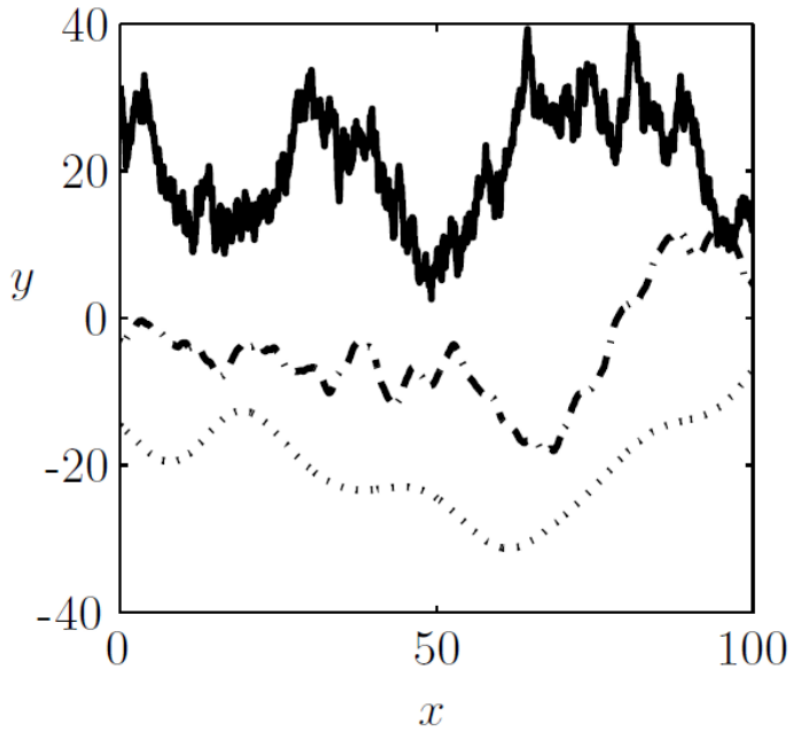
$$(\mathbf{K})_{ij} = cov^{\text{Matern}}_{\nu=5/2}(r) = \theta \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right).$$

with the parameters (same as for squared exponential)

- θ – signal variance
- ℓ – characteristic length scale

Gaussian Process covariance





from Michael Osborne: An Introduction to Fitting Gaussian Processes to Data (presentation)

Gaussian Process prediction

Making predictions

Prediction y^* in a new point $\mathbf{x}^*$ is made by adding this new point to the matrix $\mathbf{X}_N$ and vector $\mathbf{y}_N$.

This gives an $(N+1)$-dimensional Gaussian with density

$$p(\mathbf{y}_{N+1} \mid \mathbf{X}_{N+1}) = \frac{1}{\sqrt{(2\pi)^{N+1}\det(\mathbf{C}_{N+1})}} \exp(-\frac{1}{2}\mathbf{y}_{N+1}^\top \mathbf{C}_{N+1}^{-1}\mathbf{y}_{N+1})$$

where $\mathbf{C}_{N+1}$ is the covariance matrix

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^\top & \kappa + \sigma \end{pmatrix}$$

which is $\mathbf{C}_N$ extended with

- $\mathbf{k}$ – covariances between $\mathbf{x}^*$ and $\mathbf{X}_N$
- $\kappa + \sigma$ – variance of the new point itself (with added signal noise)

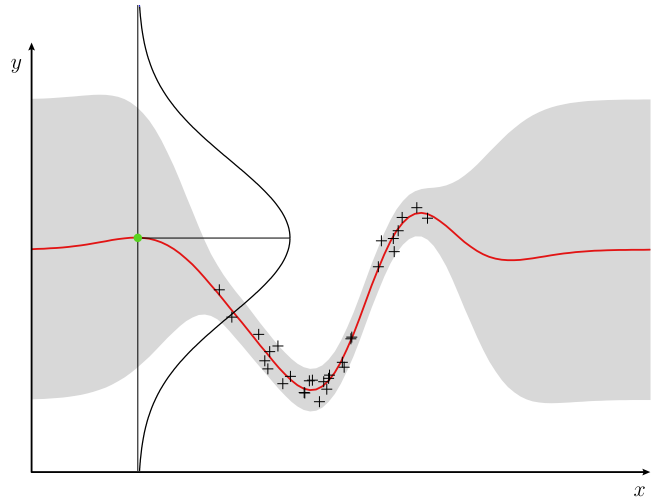
Gaussian Process prediction

Making predictions

Because $\mathbf{y}_N$ is known and the inverse $\mathbf{C}_{N+1}^{-1}$ can be expressed using inverse of the training covariance $\mathbf{C}_N{}^{-1}$,

the density in a new point marginalize to 1D Gaussian density

$$p(y^* \mid \mathbf{X}_{N+1}, \mathbf{y}_N) \propto \exp\left(-\frac{1}{2}\frac{(y^* - \hat{y}_{N+1})^2}{s^2_{y_{N+1}}}\right)$$



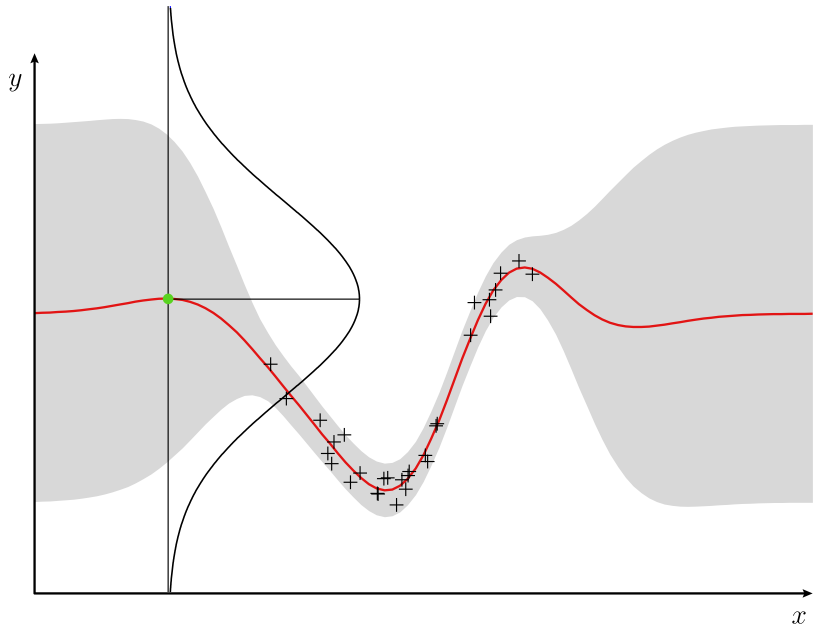
with the mean and variance given by

$$\begin{aligned} \hat{y}_{N+1} &= \mathbf{k}^\top \mathbf{C}_N{}^{-1} \mathbf{y}_N, \\ s^2_{y_{N+1}} &= \kappa - \mathbf{k}^\top \mathbf{C}_N{}^{-1} \mathbf{k}. \end{aligned}$$

Efficient Global Optimization (EGO)

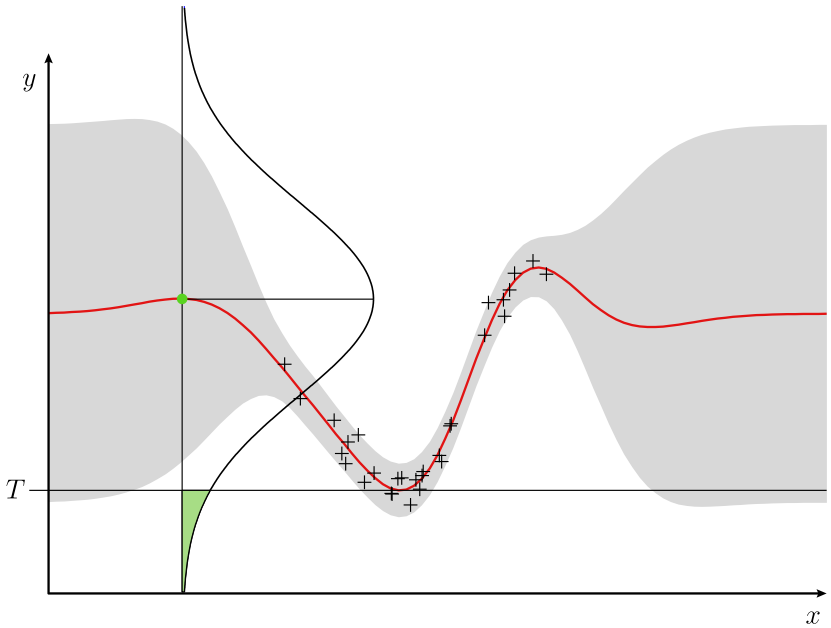
EGO

- **needed**: specific kind of surrogate model which can express uncertainty of the prediction in a new point
- EGO uses Kriging / **Gaussian processes** (GP)
- for any given input $\mathbf{x}$, it gives a probability distribution of the output value



Efficient Global Optimization (EGO)

- the resulting output for a specified input $\mathbf{x}$ is a 1-D Gaussian with
  - mean at the predicted value
  - standard deviation expressing uncertainty of this prediction



- probability of improvement (PoI) is the probability that the function value will be lower than a specified target T

$$\mathrm{PoI}_T(\mathbf{x}) = \Phi\left(\frac{T - \hat{f}(\mathbf{x})}{\hat{\sigma}(\mathbf{x})}\right) = \mathrm{P}(\hat{f}(\mathbf{x}) \leqq T)$$

Efficient Global Optimization (EGO)

EGO

1. dataset $D \leftarrow$ generate a random initial sample and evaluate it
2. build a GP/Kriging model $\hat{f}$ using D
3. $\mathbf{x}_{\max} \leftarrow$ maximize $\mathrm{PoI}(\mathbf{x})$ (from the GP model)
4. evaluate $\mathbf{x}$
5. $D = D \cup \{\mathbf{x}\}$
6. repeat from step 2

maximizing the PoI is a way of balancing between exploration of the input space and exploitation of the local optima

Model Guided Sampling Optimization

main idea of MGSO:

- consider the $\mathrm{PoI}(\mathbf{x})$ to be a function proportional to a probability density
- sample this density to get a population of candidate solutions (like EDAs)

motivation: not getting trapped in local minima while exploring the search space

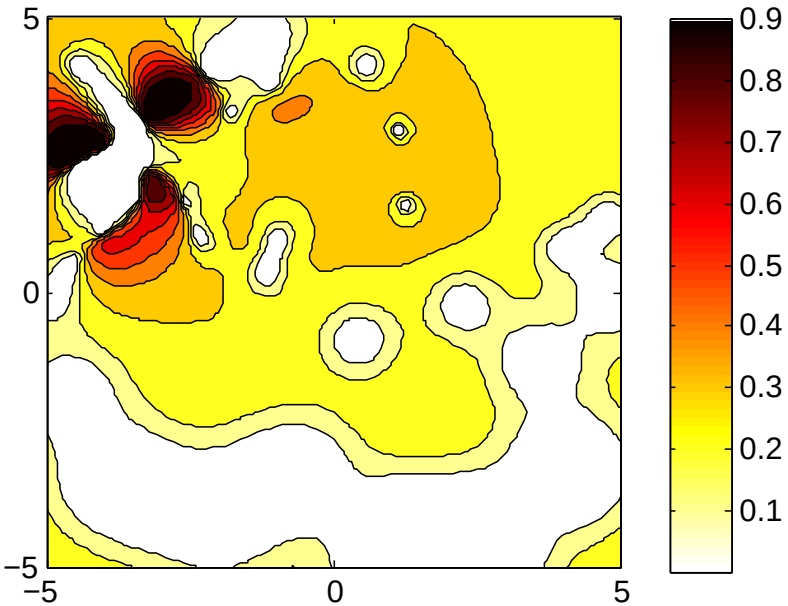
Model Guided Sampling Optimization

EGO MGSO

1. dataset $D \leftarrow$ generate a random initial sample and evaluate it
2. build a GP model $\hat{f}$ using D
3. $\mathbf{x}_{\max} \leftarrow$ maximize $\mathrm{PoI}(\mathbf{x})$ (from the GP model)
4. :)
5. sample the $\mathrm{PoI}(\mathbf{x})$ resulting in a new population P
6. find the minimum $\mathbf{x}_{\min} = \arg\min \hat{f}(\mathbf{x})$ of the GP and add to P
7. evaluate $\mathbf{x}$
8. $D = D \cup \{\mathbf{x}\}$
9. evaluate all $\mathbf{x} \in P$
10. $D = D \cup P$
11. repeat from step 2

Probability of improvement

PoI of 2D Rastrigin, $N = 40$

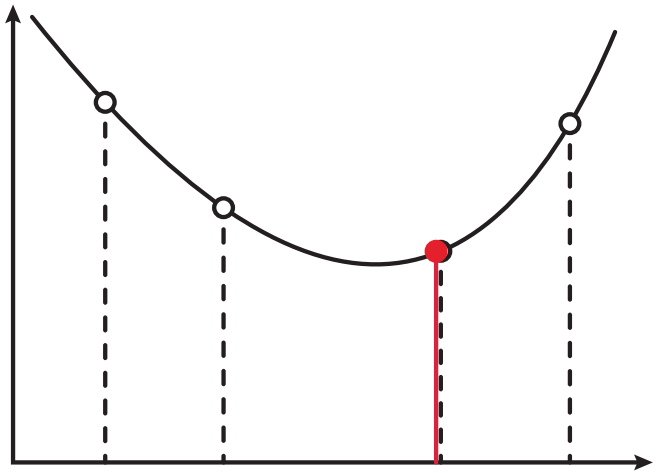


$$\mathrm{PoI}_T(\mathbf{x}) = \Phi\left(\frac{T - \hat{f}(\mathbf{x})}{\hat{\sigma}(\mathbf{x})}\right) = \mathrm{P}(\hat{f}(\mathbf{x}) \leqq T)$$

- no explicit formula for $\mathrm{PoI}(\mathbf{x}, T, \hat{f})$
- improvement in a new point $\mathbf{x}$ is probable when
  - not many samples around, i.e. large σ at $\mathbf{x}$
  - promising area is searched, i.e. low $f(\mathbf{x})$
- different way of generating new points
  - EGO: find the maximum of PoI $\rightarrow$ **one** solution
  - MGSO: sample the PoI $\rightarrow$ **multiple** solutions

Implementation issues

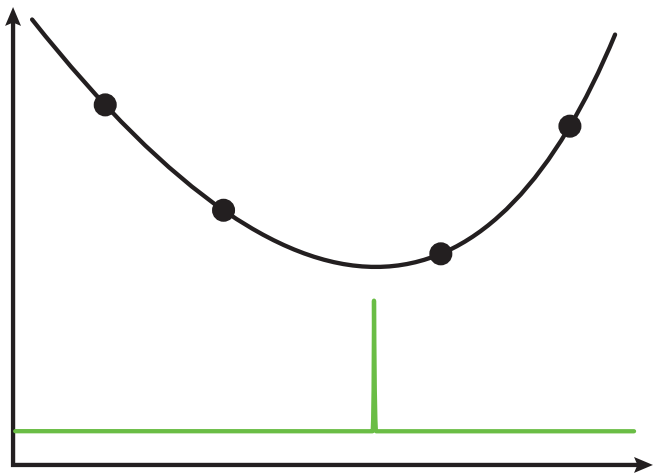
Numerical instability



- adding a new sampled point $(\mathbf{x}', f(\mathbf{x}'))$ can cause the Gaussian process' covariance matrix be close to semi-positive indefinite
- such points are **rejected** already during sampling and new points are sampled instead

Implementation issues

Degeneration of PoI

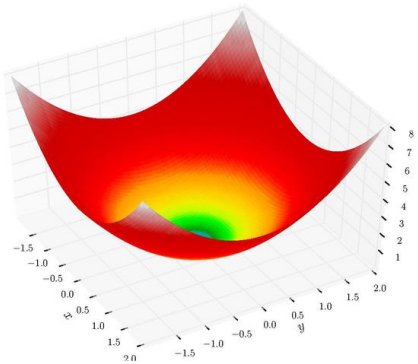


- PoI degenerates close to a discrete distribution in later phases of the optimization
- happens when the input space is well explored
- partially solved by cropping the input space to a region around the **so-far optimum** and rescaling the input space to get better numerical resolution for sampling

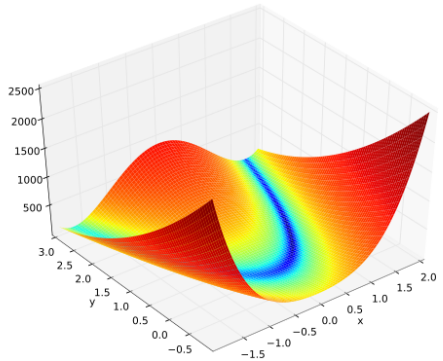
[... to be continued ...]

Experimental results on BBOB

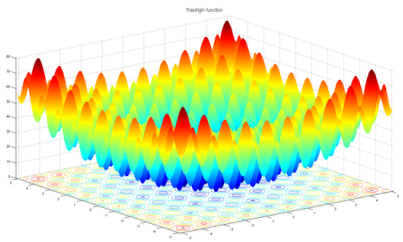
- MGSO was also tested on three benchmark functions from the BBOB testbed



sphere



Rosenbrock

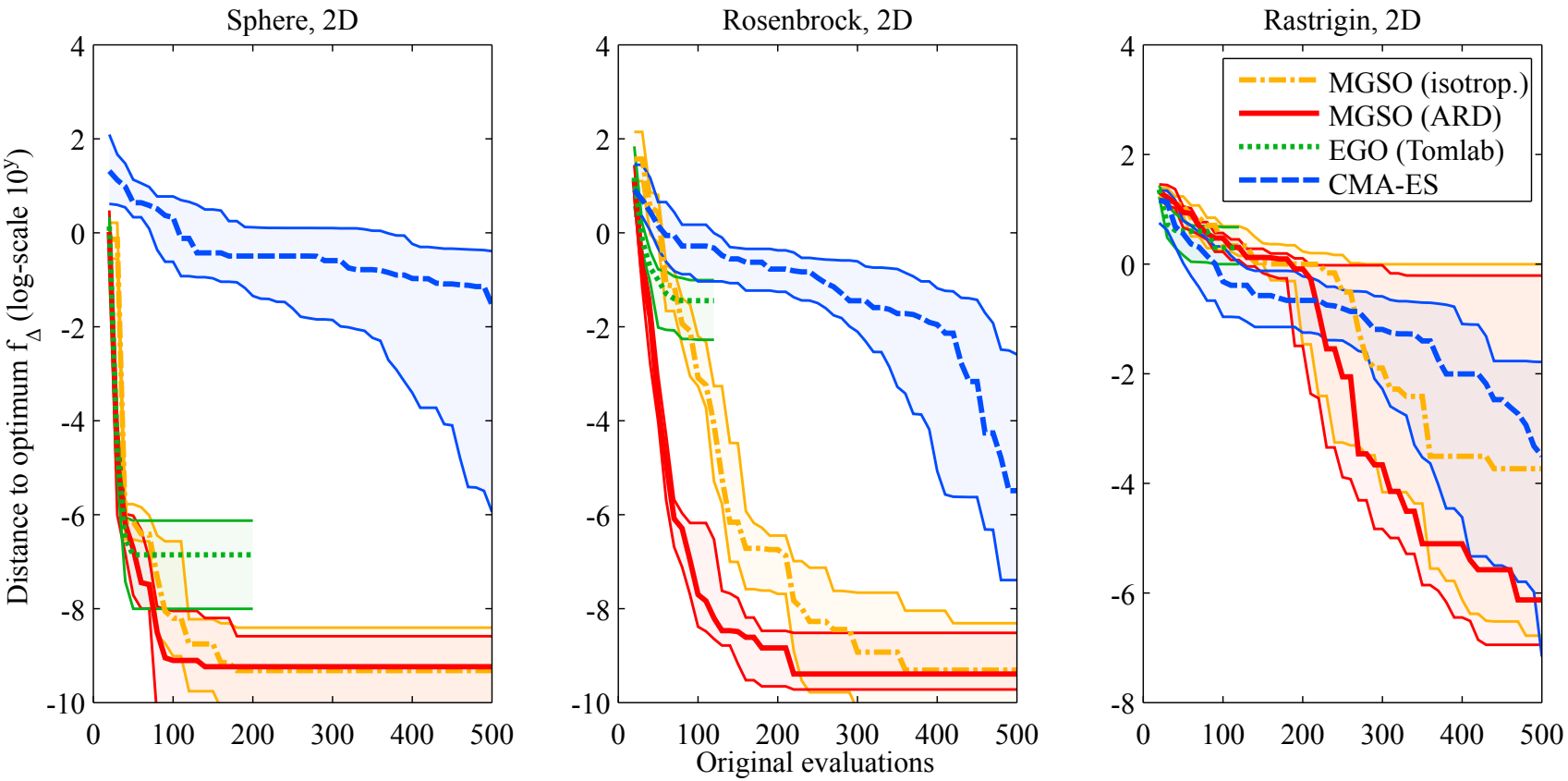


Rastrigin

(CC) Wikimedia Commons

- three dimensionalities: 2D, 5D, 10D
- compared to CMA-ES and Tomlab's EGO implementation

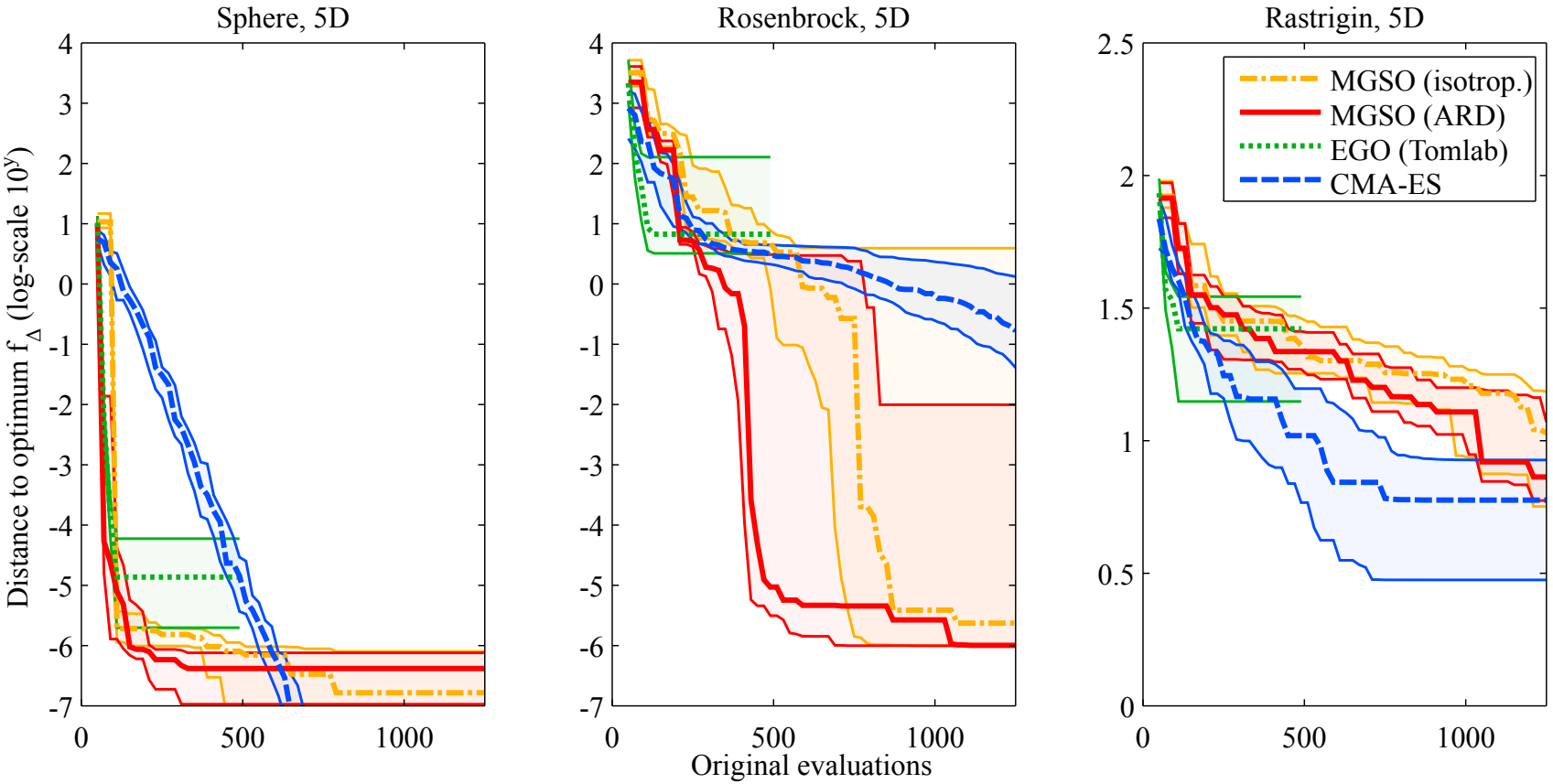
Results in 2-D



medians and quartiles of the best-found fitness from 15 runs

MGSO, CMA-ES, EGO, older MGSO without ARD

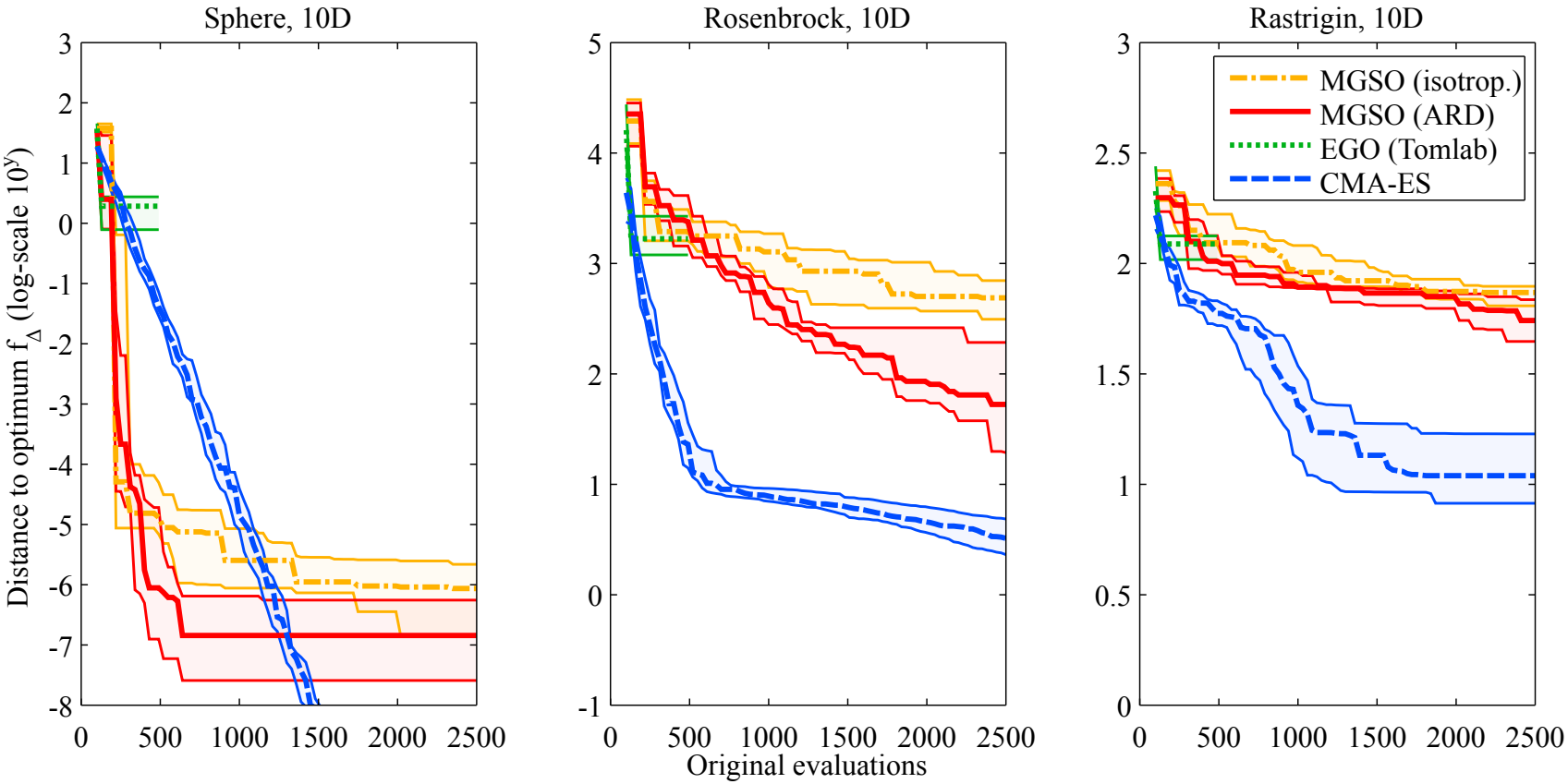
Results in 5-D



medians and quartiles of the best-found fitness from 15 runs

MGSO, CMA-ES, EGO, older MGSO without ARD

Results in 10-D



medians and quartiles of the best-found fitness from 15 runs

MGSO, CMA-ES, EGO, older MGSO without ARD

Stochastic search of Evolutionary algorithms

Stochastic black box search

initilize distribution parameters θ
set population size $\lambda \in \mathbb{N}$
while not terminate

1. sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. evaluate $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ on f
3. update parameters θ

(A. Auger, Tutorial CMA-ES, GECCO 2013)

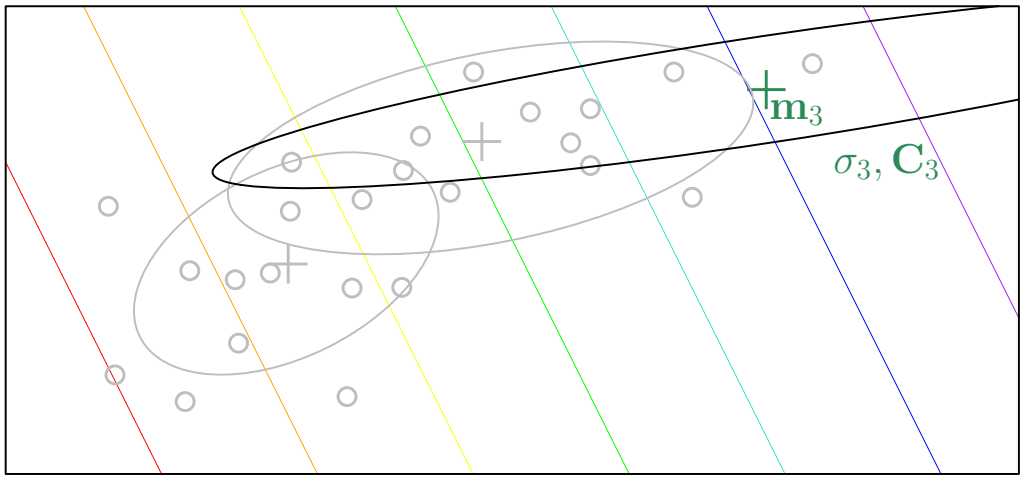
- schema of most of the evolutionary strategies (and EDA algorithms)
- as well as CMA-ES (Covariance Matrix Adaptation ES) – current state of the art in continuous optimization

The CMA-ES

Input: $\mathbf{m} \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \in \mathbb{N}$
Initialize: $\mathbf{C} = \mathbf{I}$ (and several other parameters)
Set the weights $w_1, \ldots w_\lambda$ appropriately

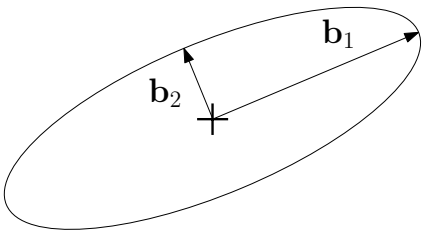
while not terminate

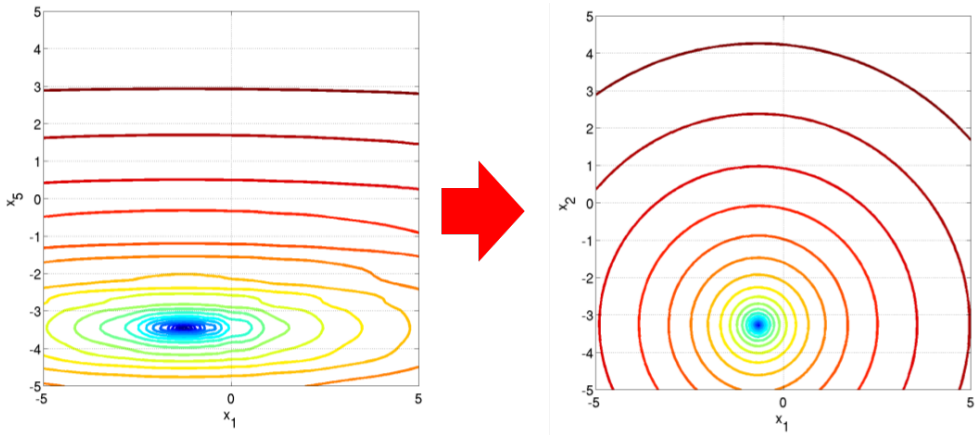
1. $\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i, \quad \mathbf{y}_i \sim N(\mathbf{0}, \mathbf{C}), \quad$ for $i = 1, \ldots, \lambda$ sampling
2. $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \, \mathbf{x}_{i:\lambda} = \mathbf{m} + \sigma \mathbf{y}_w \quad$ where $\mathbf{y}_w = \sum_{i=1}^{\mu} w_i \, \mathbf{y}_{i:\lambda}$ update mean
3. update $\mathbf{C}$
4. update step-size σ



Covariance matrix adaptation

- eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components – the principle axes of the mutation ellipsoid
- CMA-ES learns and updates a new Mahalanobis metric
- successively approximates the inverse Hessian on quadratic functions
  – transforms ellipsoid function into sphere function
  – it somehow holds for other functions, too (up to some degree)





Is CMA-ES the best for everything?

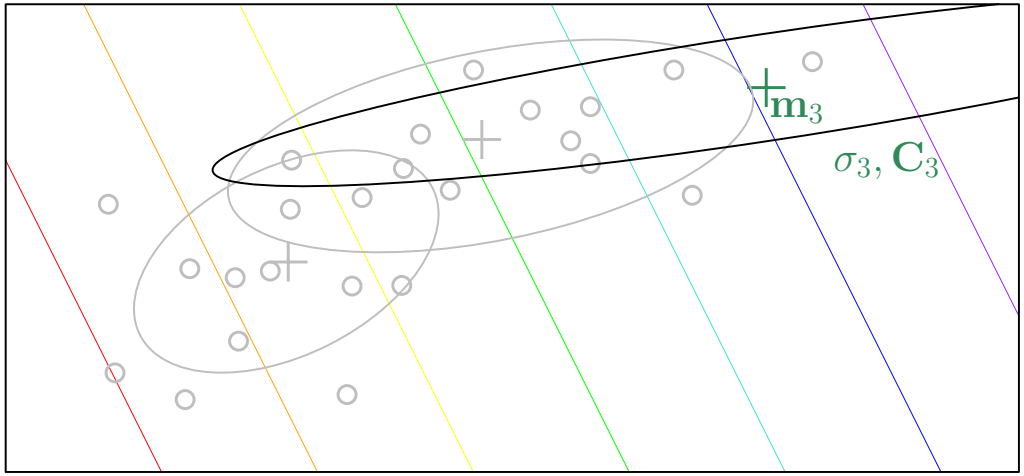
- CMA-ES is state-of-the-art optimization algorithm, especially for rugged and ill-conditioned objective functions
- however, **not the fastest** if we can afford only very few objective function evaluations
- what we have already seen: use a surrogate model!
- however, original evaluated solutions are available only along the search path
- solution: construct local surrogate models

The Surrogate CMA-ES

Input: $\mathbf{m} \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \in \mathbb{N}$
Initialize: $\mathbf{C} = \mathbf{I}$ (and several other parameters)
Set the weights $w_1, \ldots w_\lambda$ appropriately

while not terminate

1. $\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i, \quad \mathbf{y}_i \sim N(\mathbf{0}, \mathbf{C}), \quad$ for $i = 1, \ldots, \lambda$ sampling
2. evaluate with the original fitness & build a model
3. evaluate with the model fitness
4. update $\mathbf{m}, \mathbf{C}, \sigma$



The Surrogate CMA-ES

Input: g (generation), $f_{\mathcal{M}}$ (model), $\mathcal{A}$ (archive), n_{REQ}, σ, λ, $\mathbf{m}$, $\mathbf{C}$

$\mathbf{x}_k \sim \mathcal{N}\left(\mathbf{m}, \sigma^2\mathbf{C}\right) \quad k = 1, \ldots, \lambda$ {*CMA-ES sampling*}

if g is original-evaluated **then**

  $y_k \leftarrow f(\mathbf{x}_k) \quad k = 1, \ldots, \lambda$ {*fitness evaluation*}

  $\mathcal{A} = \mathcal{A} \cup \{(\mathbf{x}_k, y_k)\}_{k=1}^{\lambda}$

  if $|\mathbf{X}| \geq n_{\text{REQ}}$ **then**

    $\mathbf{X} \leftarrow$ TransformToTheEigenvectorBasis($\mathbf{X}$, σ, $\mathbf{C}$)

    $f_{\mathcal{M}} \leftarrow$ trainModel($\mathbf{X}, \mathbf{y}$)

  end if

else

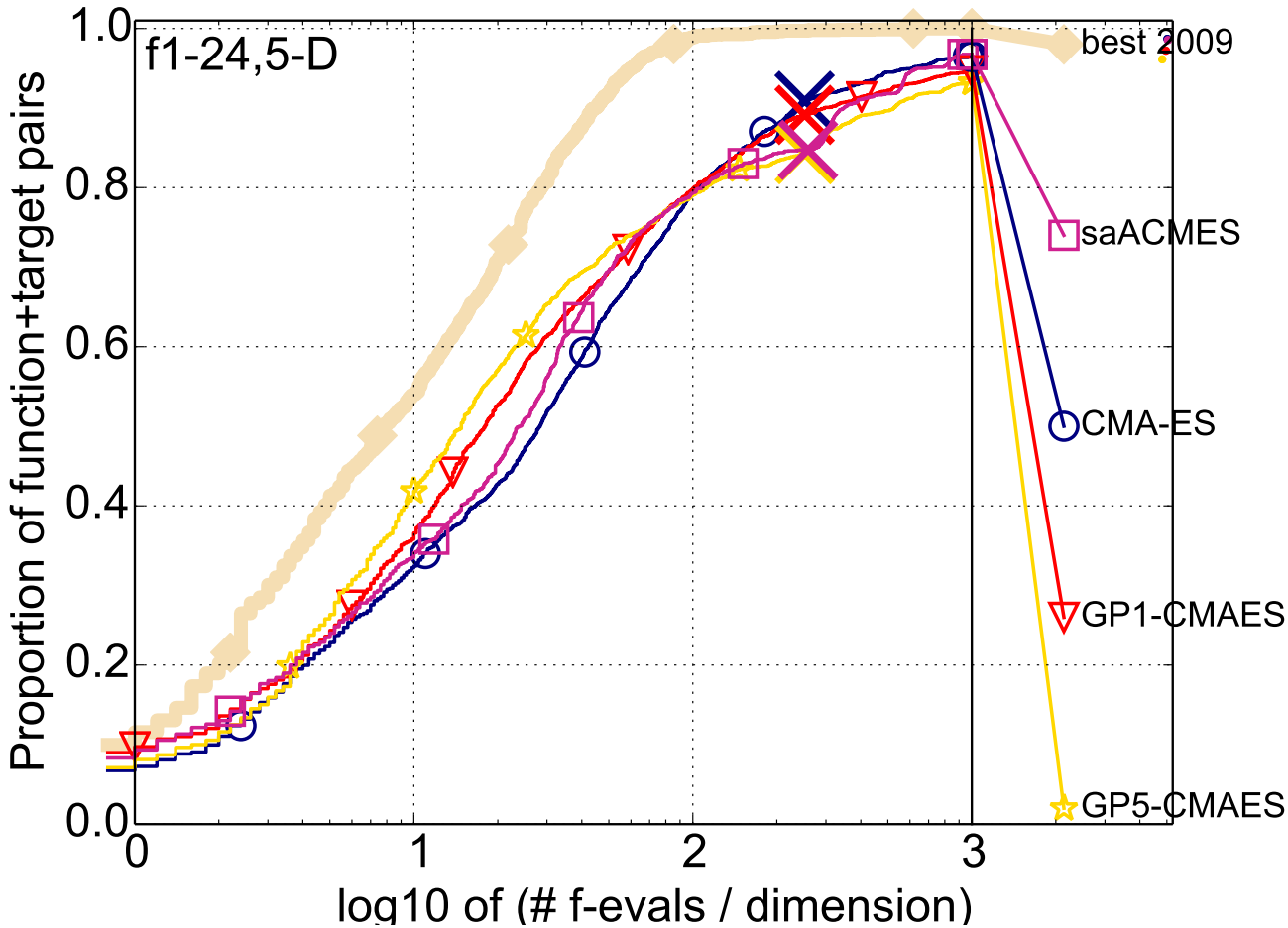
  $\mathbf{X} \leftarrow$ TransformToTheEigenvectorBasis($\mathbf{X}$, σ, $\mathbf{C}$)

  $y_k \leftarrow f_{\mathcal{M}}(\mathbf{x}_k) \quad k = 1, \ldots, \lambda$ {*model evaluation*}

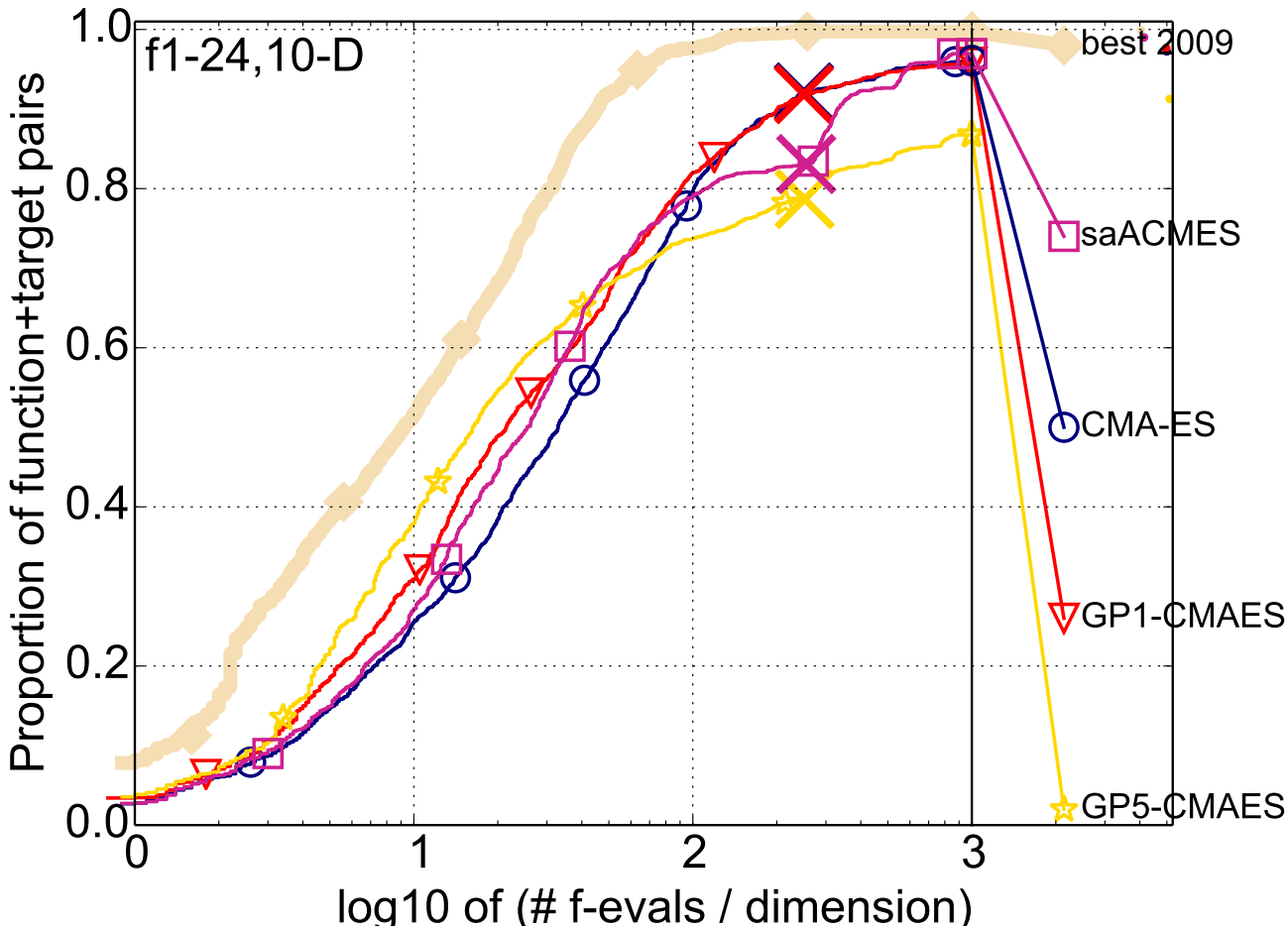
end if

[... to be continued ...]

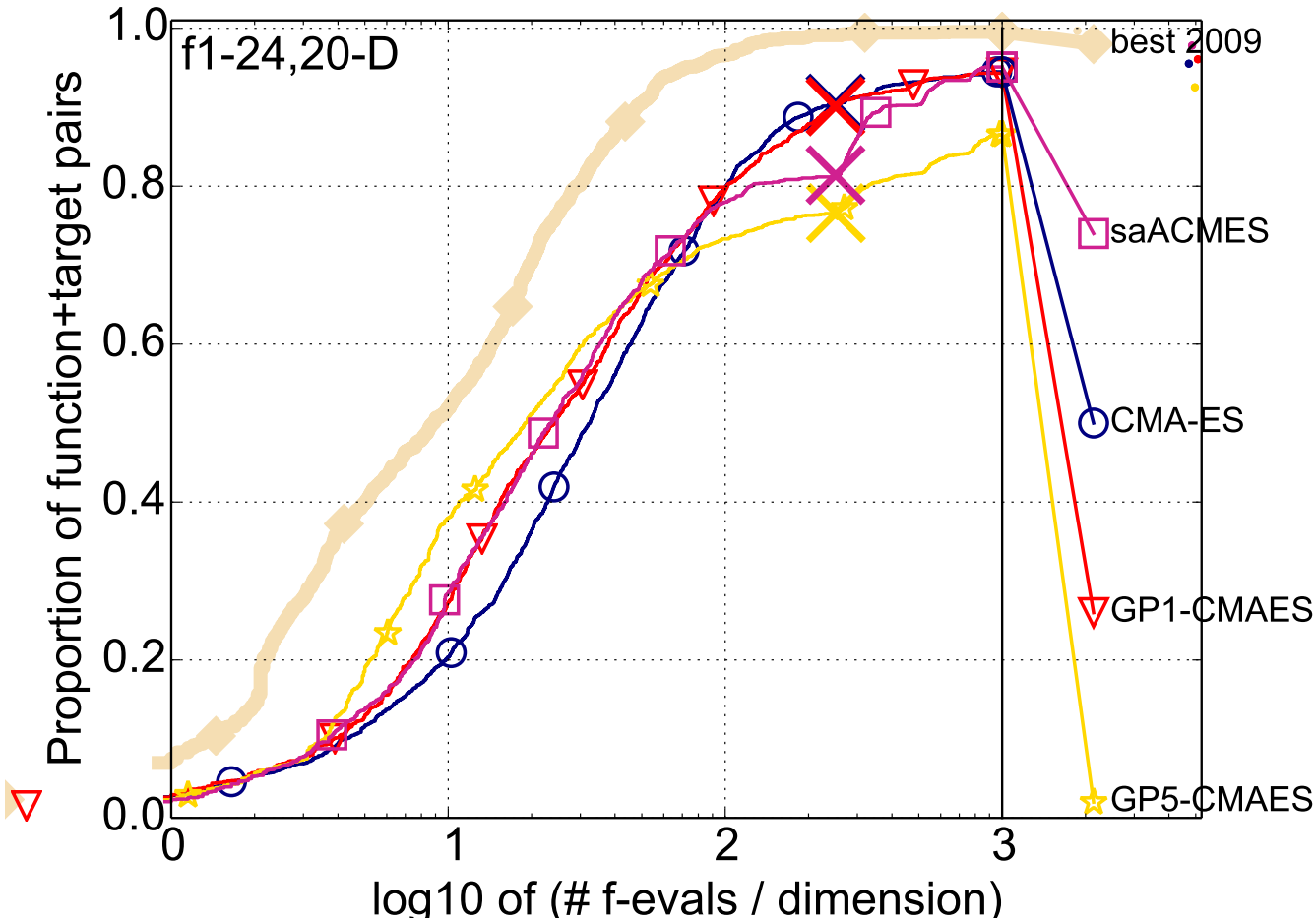
Experimental results on BBOB (5 D)



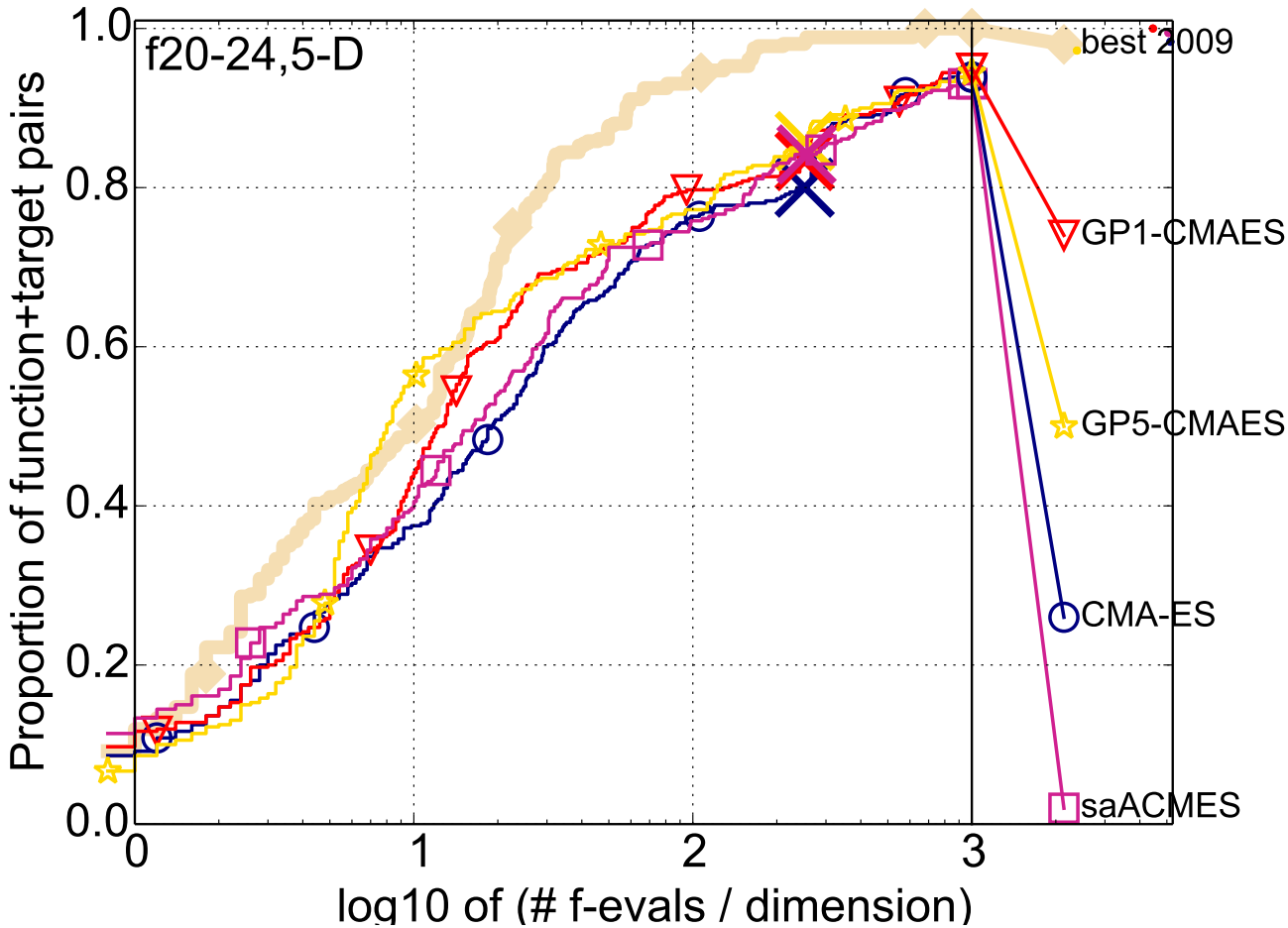
Experimental results on BBOB (10 D)



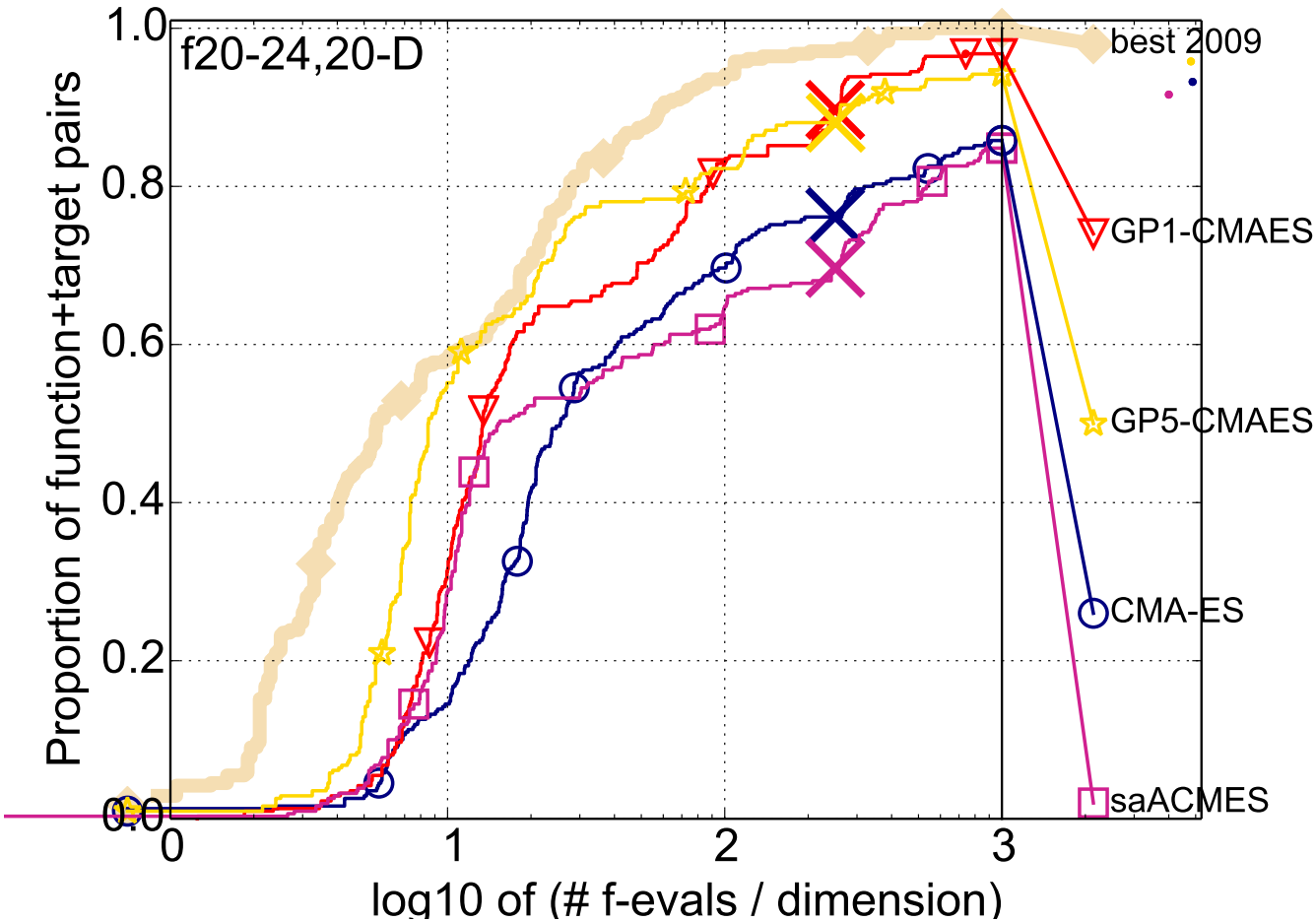
Experimental results on BBOB (20 D)



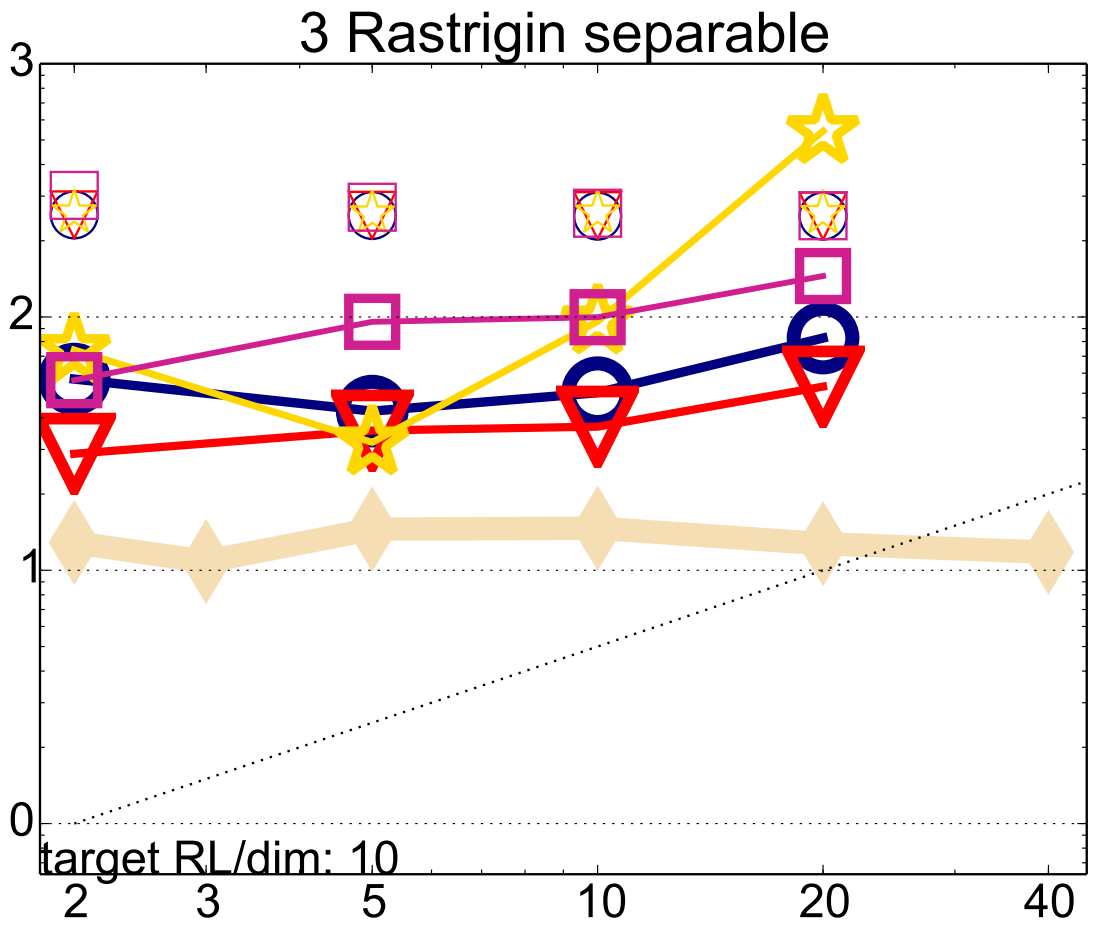
The best results on subset of BBOB (5 D)



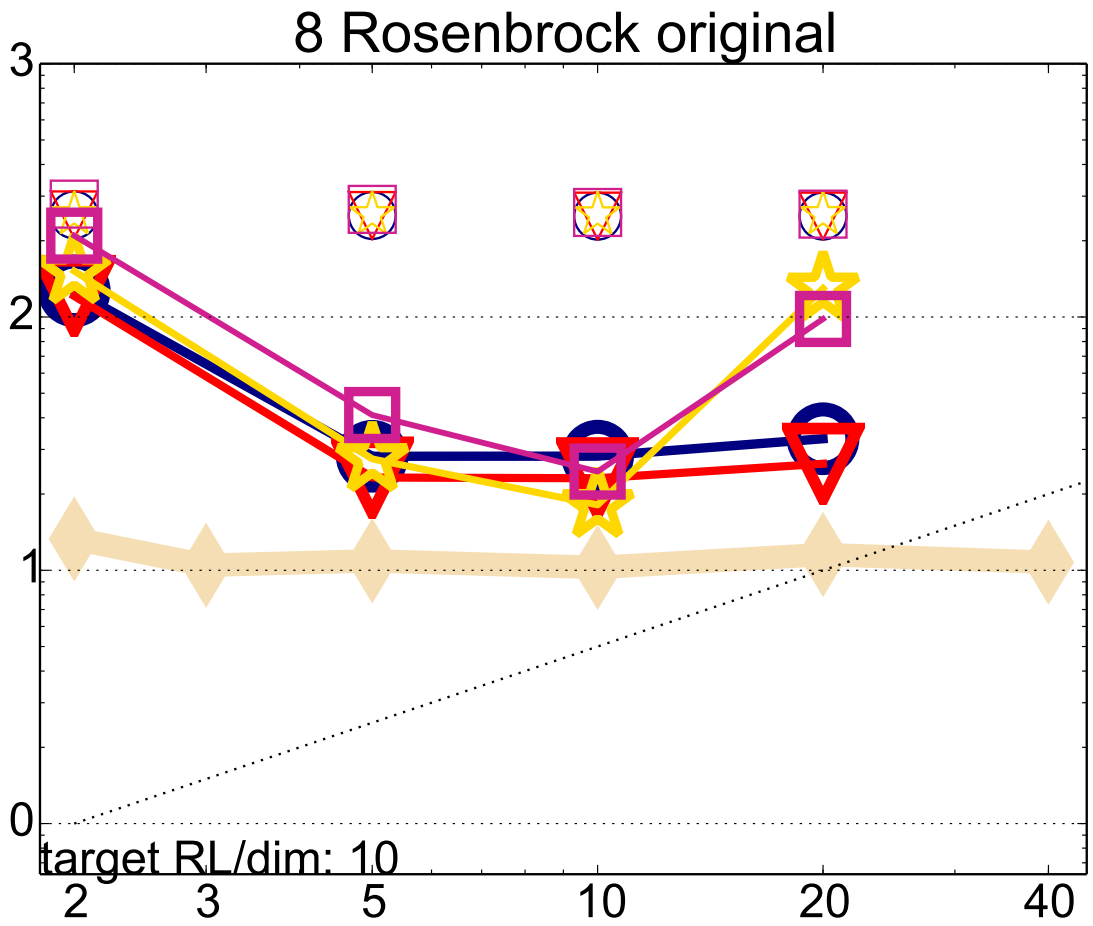
The best results on subset of BBOB (20 D)



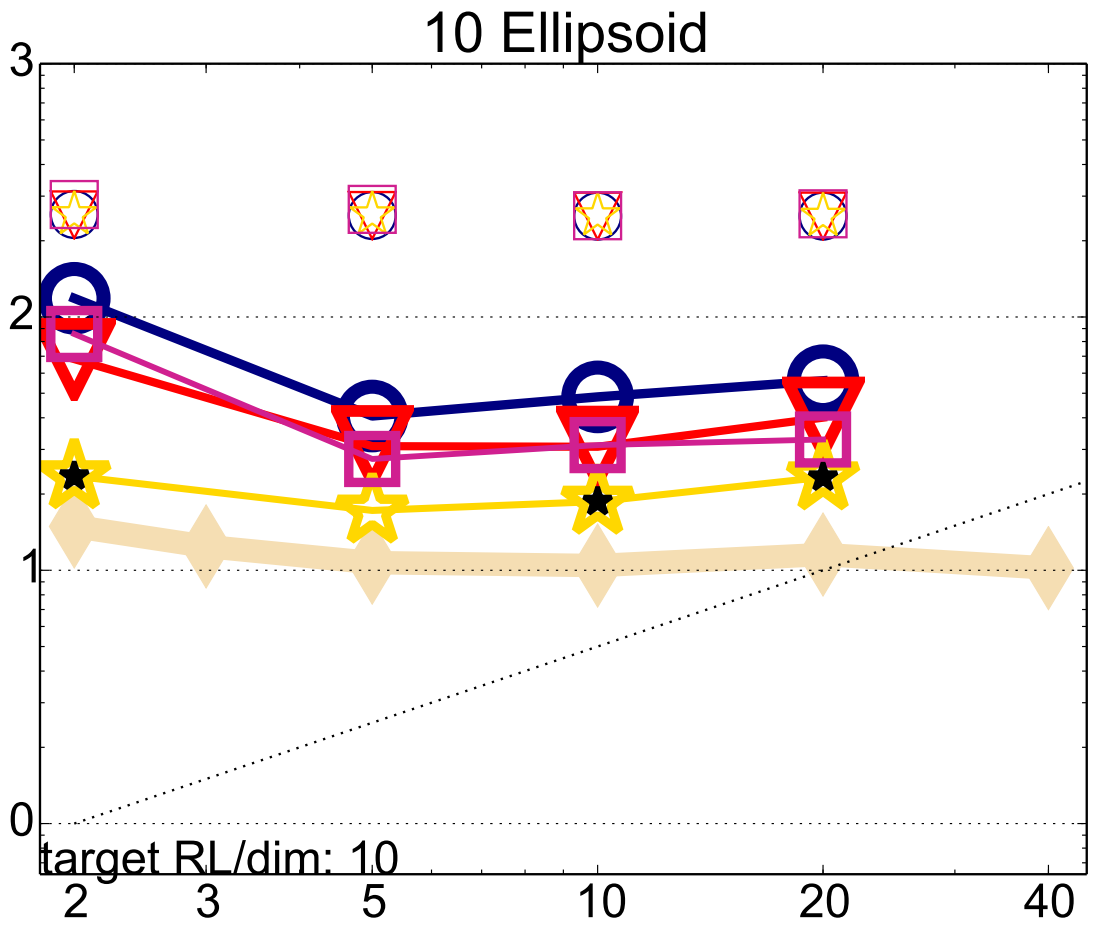
Results on #3 Rastrigin separable



Results on #8 Rosenbrock



Results on #10 Ellipsoid function



Thank you!

bajer@cs.cas.cz

Tested approaches:

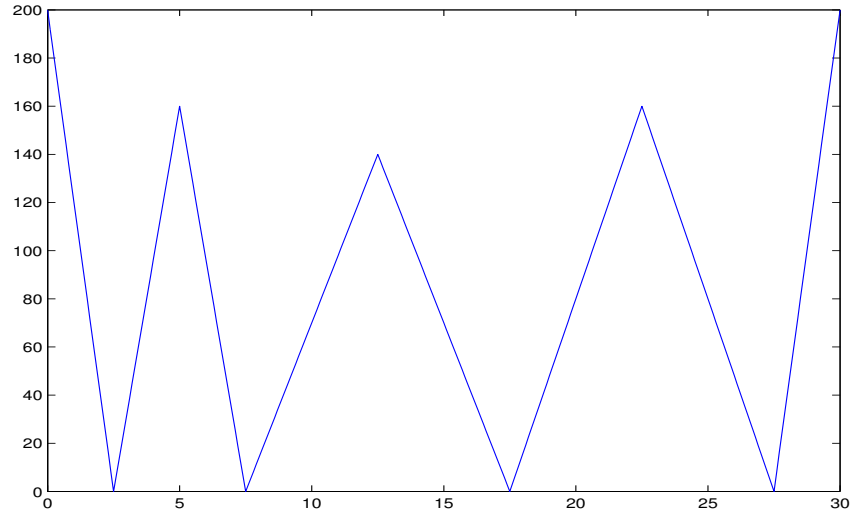
- Model Guided Sampling Optimization (MGSO),
- CMA-ES with Gaussian process as a surrogate model (S-CMA-ES).

Testing functions

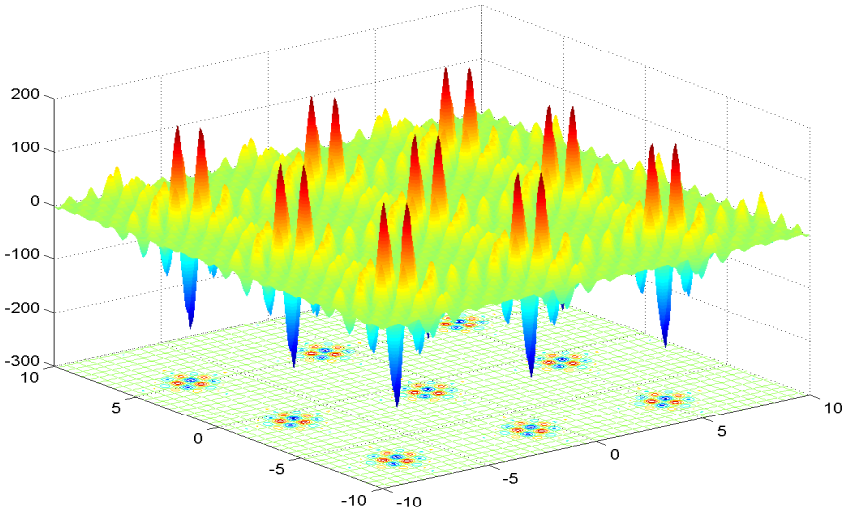
Testing was performed on 20 versions of 12 multimodal fitness functions from CEC 2013 competition:

- Characterized by a high number of local optima;
- Some functions high-dimensional (up to 20D).

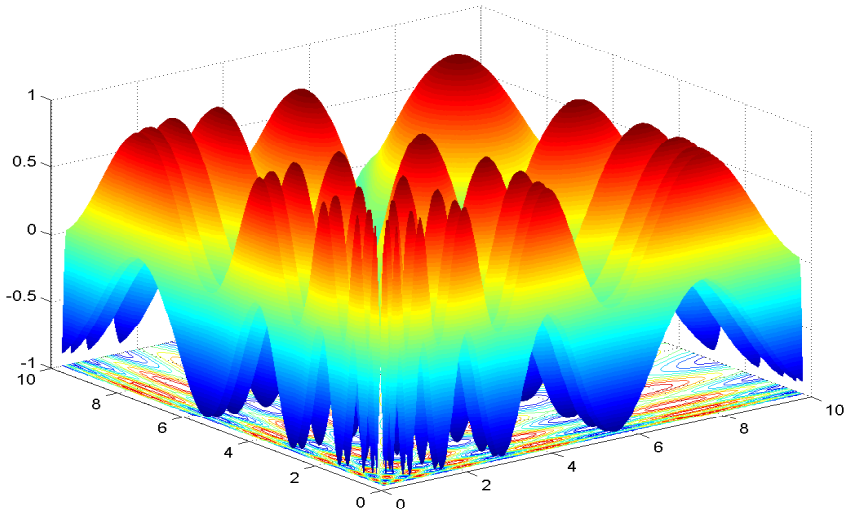
$\rightarrow$ difficult optimalization task



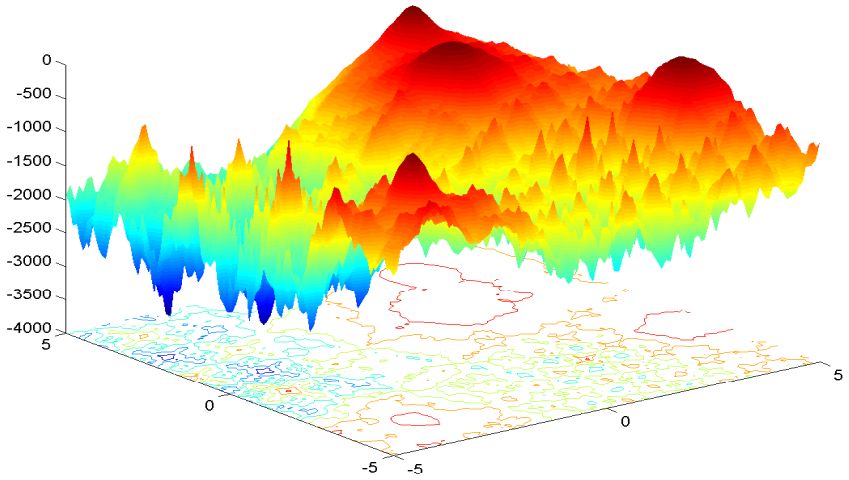
(a) Five-Uneaven-Peak Trap



(b) Inverted Shubert



(c) Vincent



(d) Composite Function 4

Part I

MGSO testing

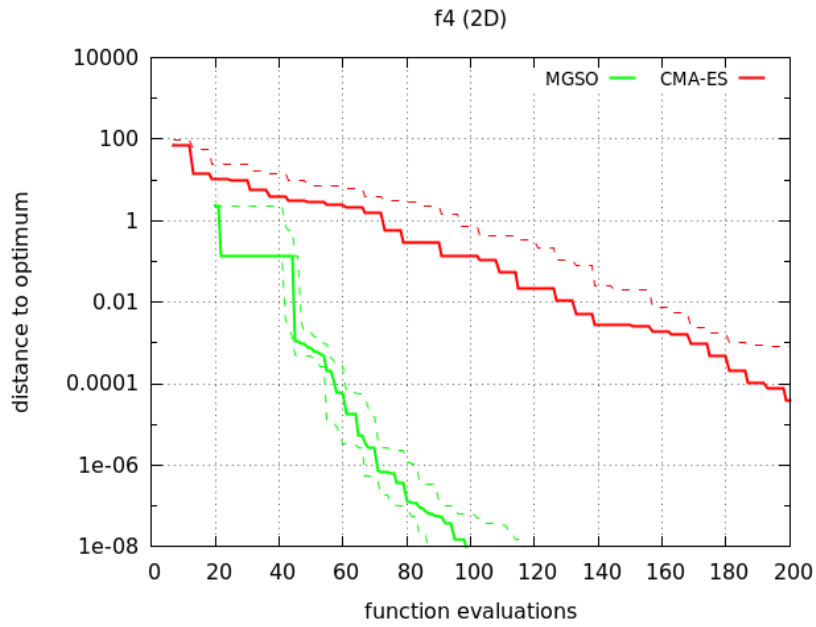
Examined parameters

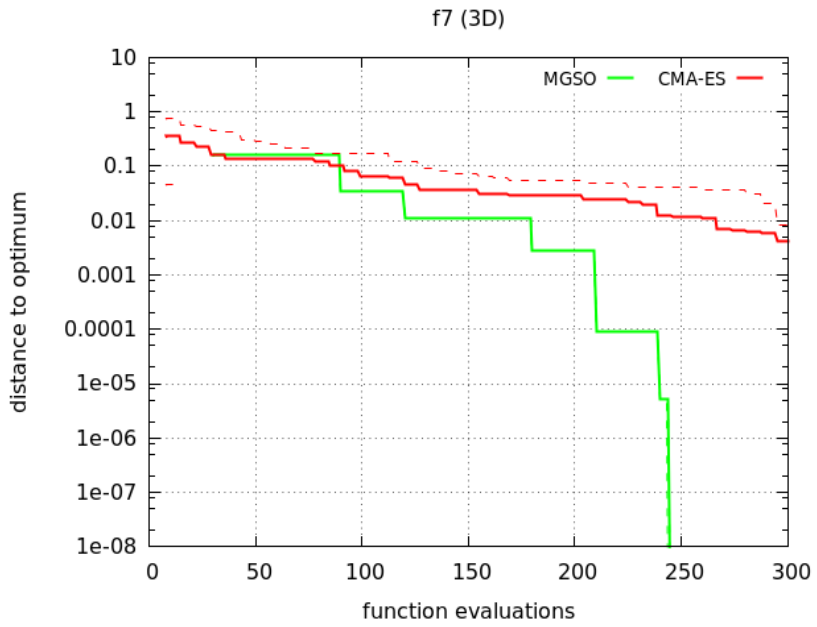
Two covariance functions:

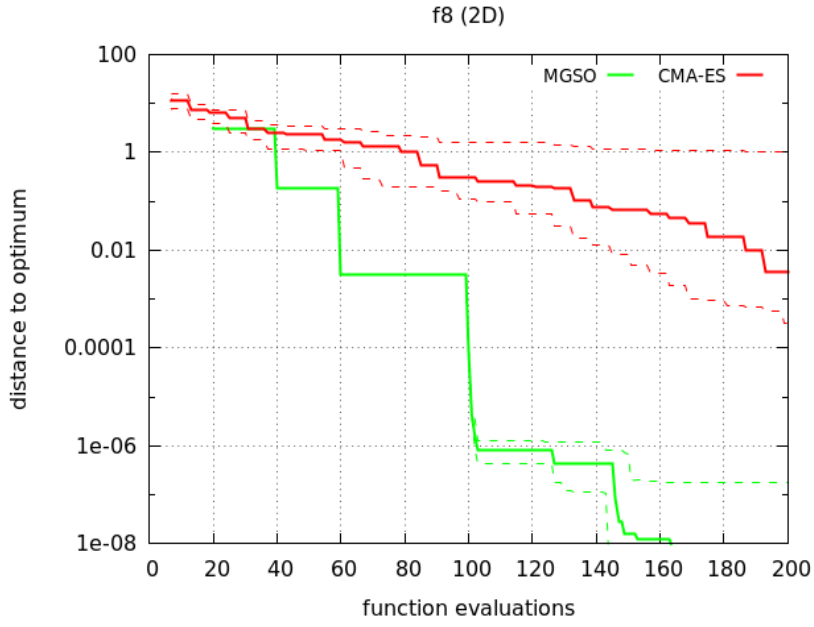
- $\mathbf{K}_{\text{SE}}^{\text{iso}}$ – isometric squared exponential,
- $\mathbf{K}_{\text{SE}}^{\text{ard}}$ – squared exponential with automatic relevance determination.

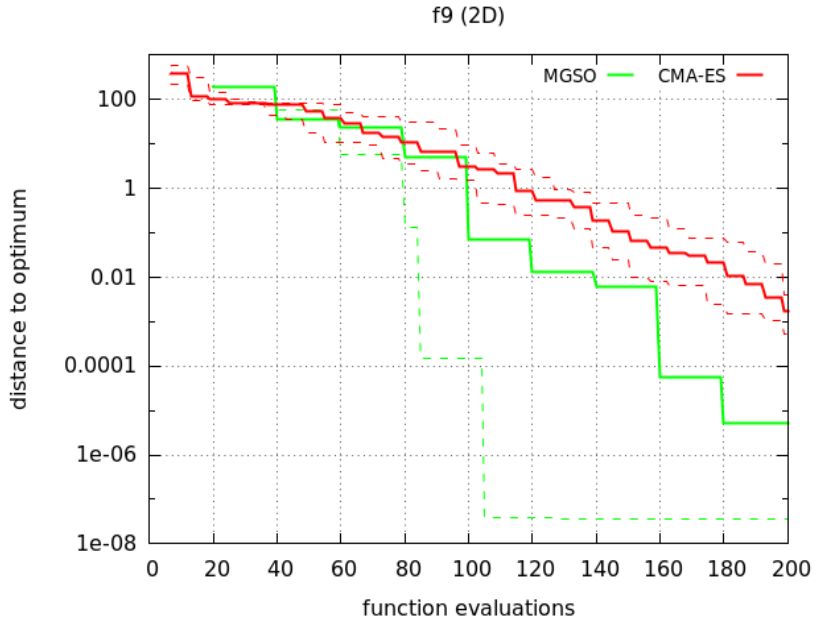
Observations

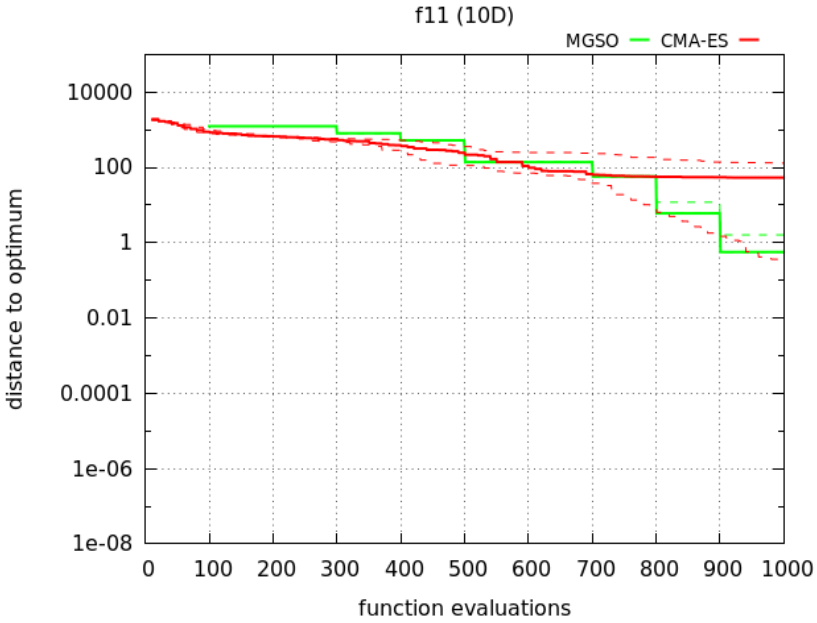
- Both covariance functions had similar effect on the performance.
- MGSO achieved better results than CMA-ES in a vast majority of functions.
- The considerable performance was achieved in the case of most low-dimensional (2D-5D) functions.
- In the case of high-dimensional functions (10D-20D), the speed-up was comparable to CMA-ES.

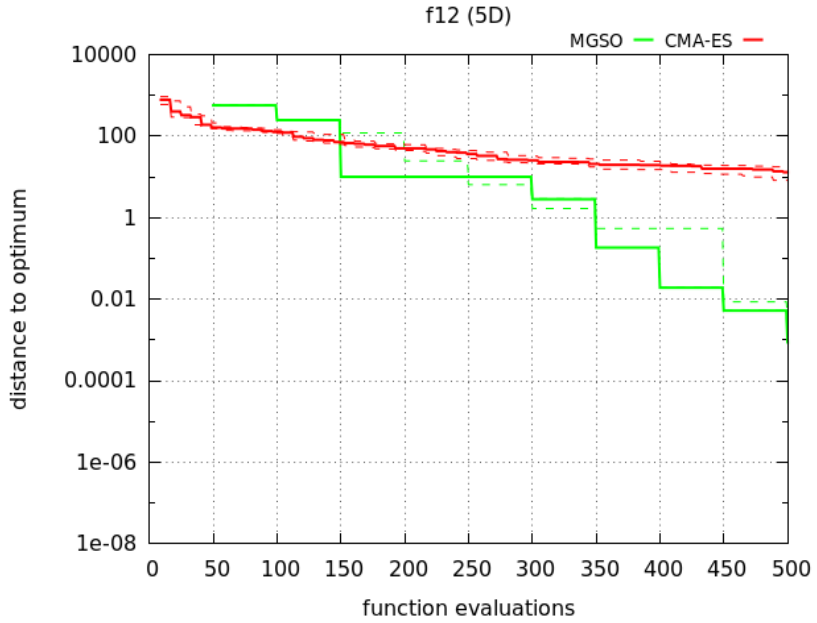

f4 (2D)
MGSO
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200

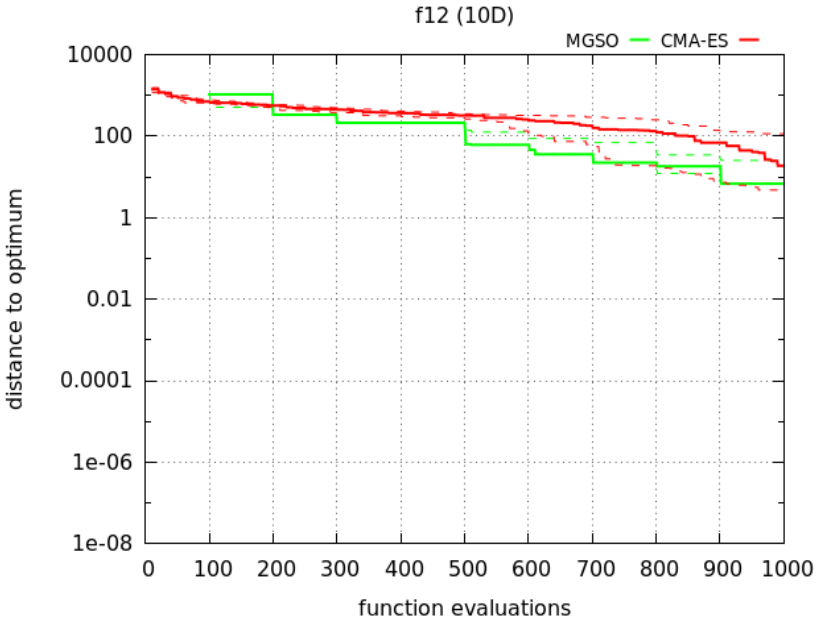

f7 (3D)
MGSO
CMA-ES
distance to optimum
function evaluations
10
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0
50
100
150
200
250
300

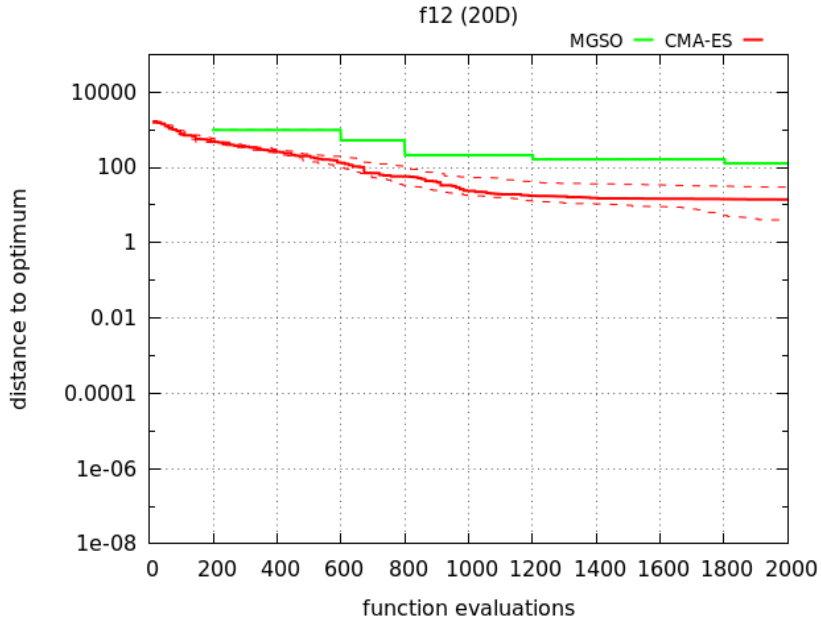

f8 (2D)
MGSO
CMA-ES
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f9 (2D)
MGSO
CMA-ES
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f11 (10D)
MGSO
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
100
200
300
400
500
600
700
800
900
1000


f12 (5D)
MGSO
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300
350
400
450
500


f12 (10D)
MGSO
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
100
200
300
400
500
600
700
800
900
1000


f12 (20D)
MGSO
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
200
400
600
800
1000
1200
1400
1600
1800
2000

Part II

S-CMA-ES testing

Examined parameters

Four covariance functions:

- $\mathbf{K}_{\text{SE}}^{\text{iso}}$ – isometric squared exponential,
- $\mathbf{K}_{\text{SE}}^{\text{ard}}$ – squared exponential with automatic relevance determination,
- $\mathbf{K}_{\text{Matérn}}$ – with $\nu = \frac{5}{2}$,
- $\mathbf{K}_{\text{Matérn}}$ – with $\nu = \frac{1}{2}$, a.k.a. exponential covariance function.

Evolution control strategies

S-CMA-ES evolution control (EC) strategies:

- individual-based,
- generation-based.

Generation-based EC strategy

Evolution control settings:

- number of consecutive generations evaluated by a model,
- multiplication factor of CMA-ES' step size.

Individual-based EC strategy

Evaluates only a part of the population using the original fitness function:

1. Pre-sample some individuals and train the model;
2. Create the extended population by sampling from the model;
3. Evaluate a fraction of the individuals from the extended population using the original fitness function;
4. Cluster the rest of the extended population and add best point to the final population;

Individual-based EC strategy

Evolution control settings:

- number of pre-sampled individuals evaluated by the fitness function (used for model training),
- size of the extended population,
- amount of points chosen from the extended population to be evaluated by the fithess function.

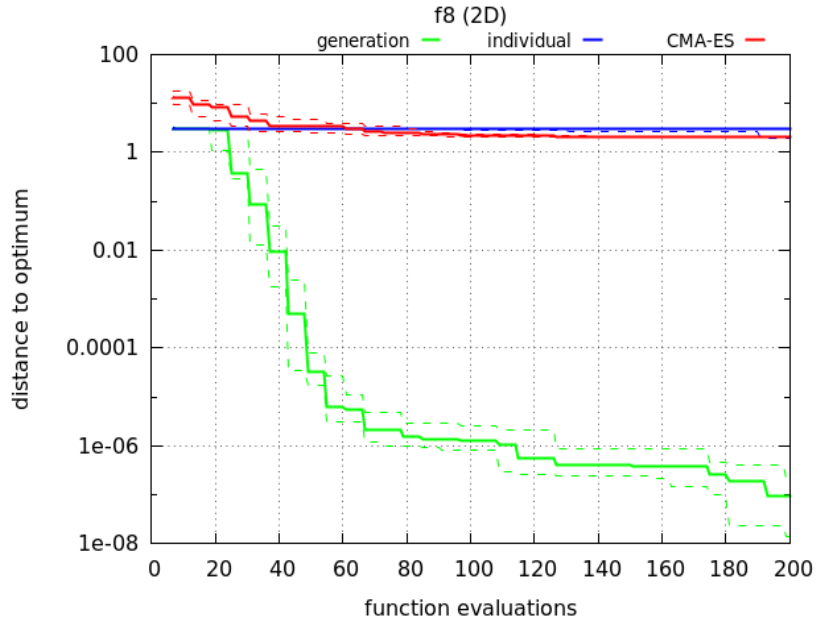
S-CMA-ES observations

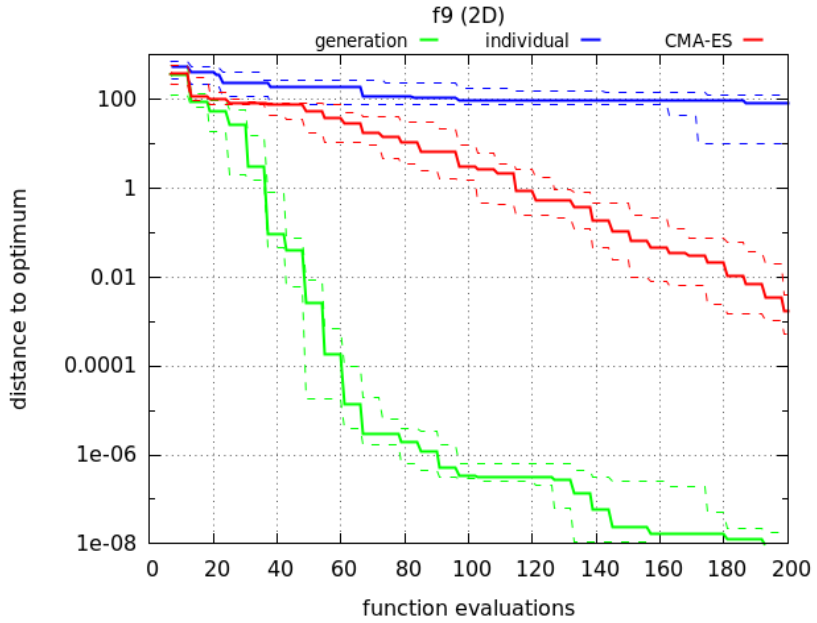
Individual-based EC strategy:

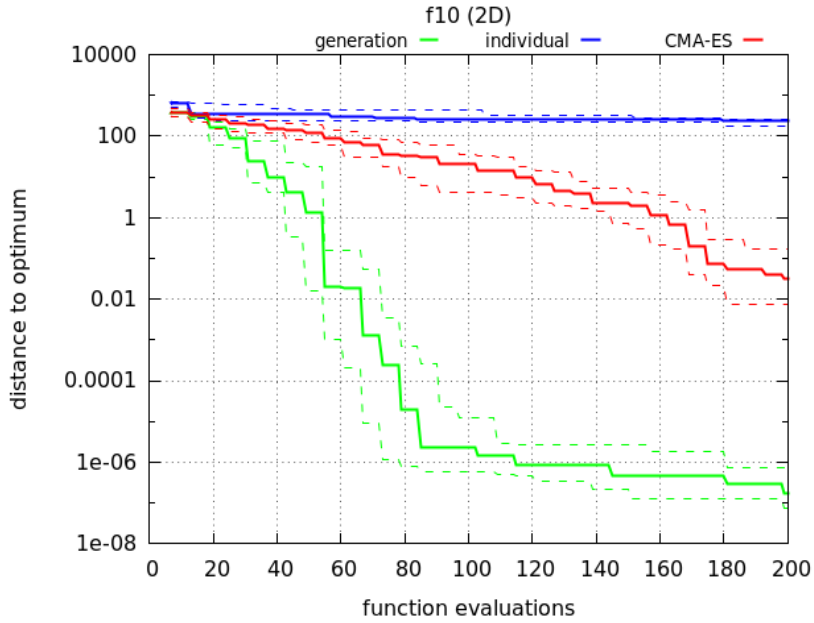
- peformed worse in the case of all functions.

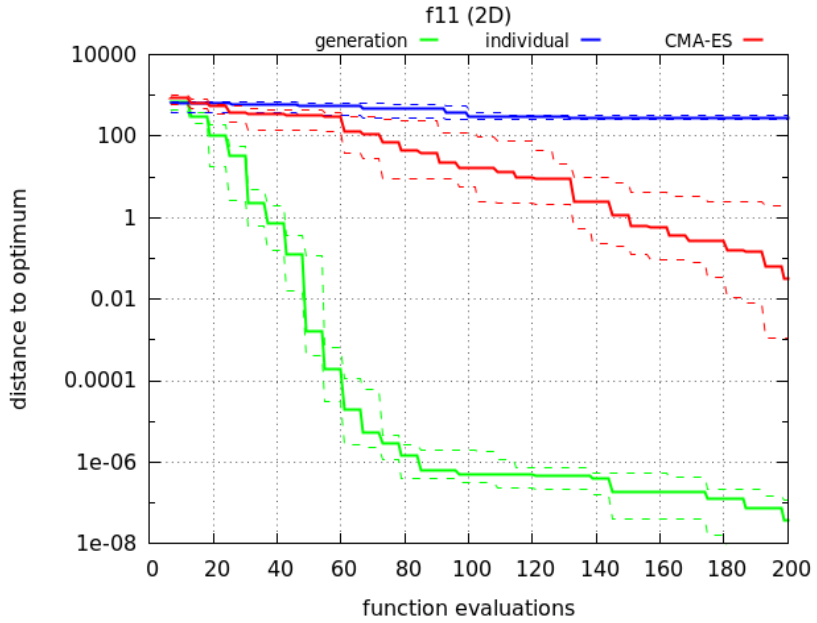
Generation-based EC strategy:

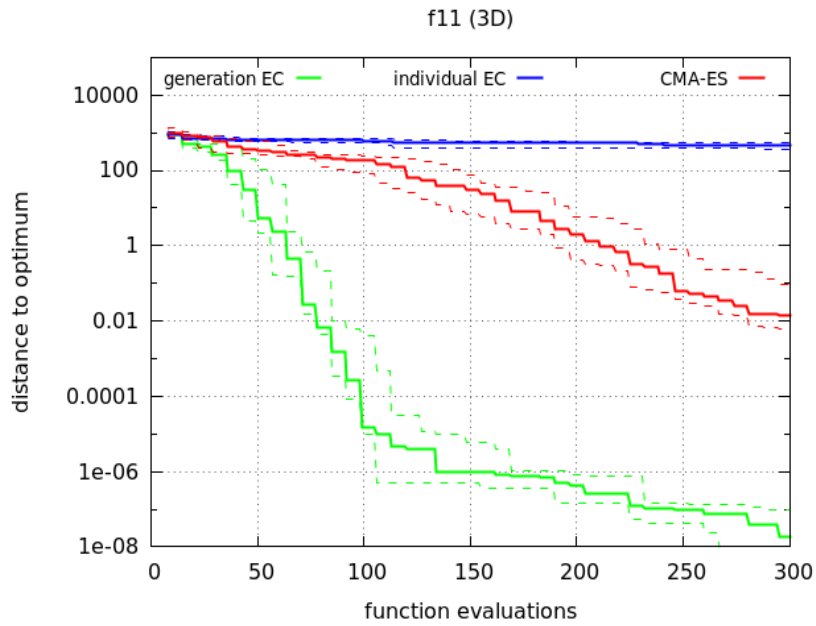
- showed the performance improvement in the case of almost all functions,
- performed better using more consequent model-evaluated generations, unmodified step size and $\mathbf{K}_{\mathrm{SE}}^{\mathrm{iso}}$ covariance function.

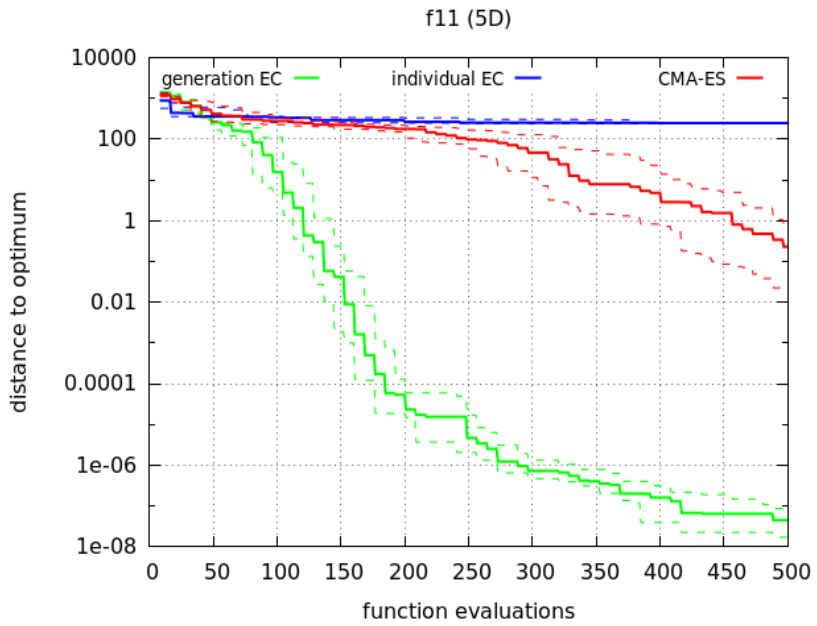

f8 (2D)
generation
individual
CMA-ES
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200

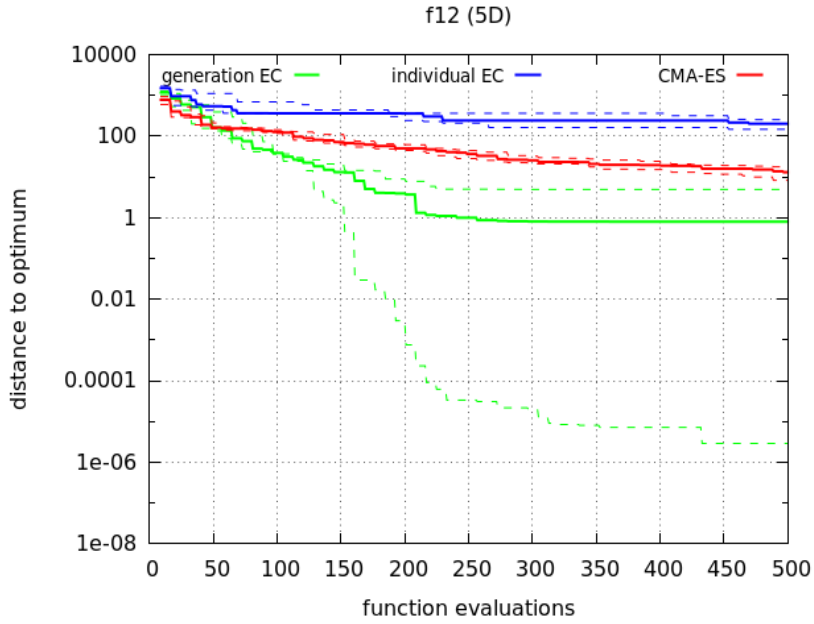

f9 (2D)
generation
individual
CMA-ES
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200

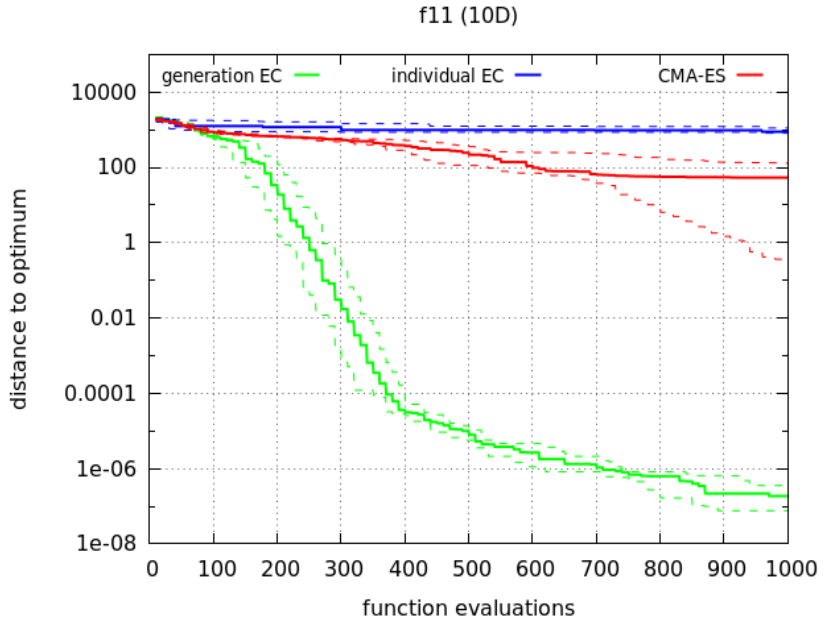

f10 (2D)
generation
individual
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f11 (2D)
generation
individual
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f11 (3D)
generation EC
individual EC
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300

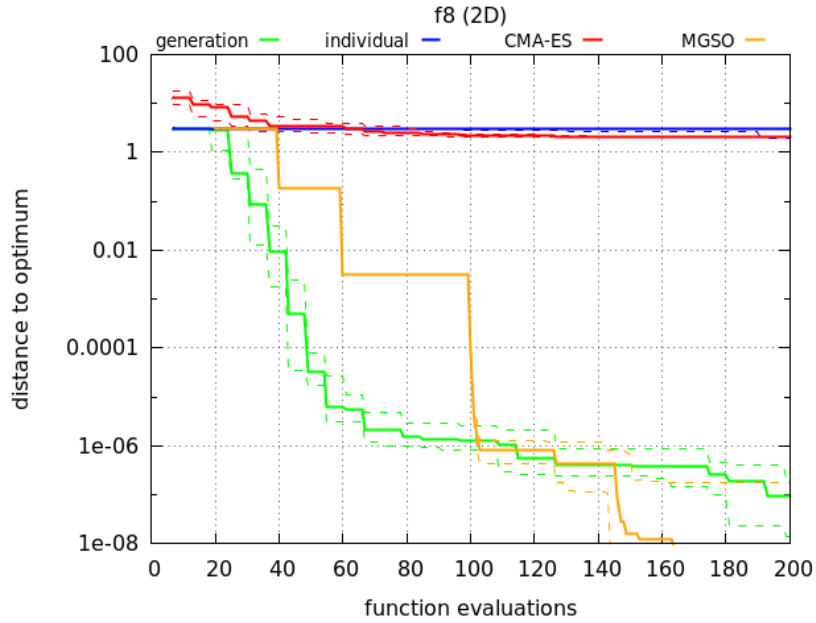

f11 (5D)
generation EC
individual EC
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300
350
400
450
500

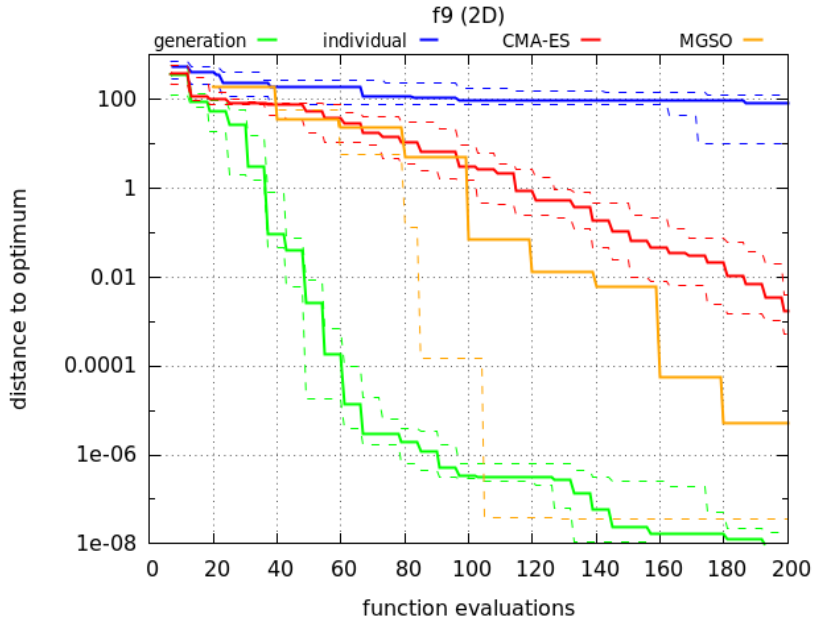

f12 (5D)
generation EC
individual EC
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300
350
400
450
500

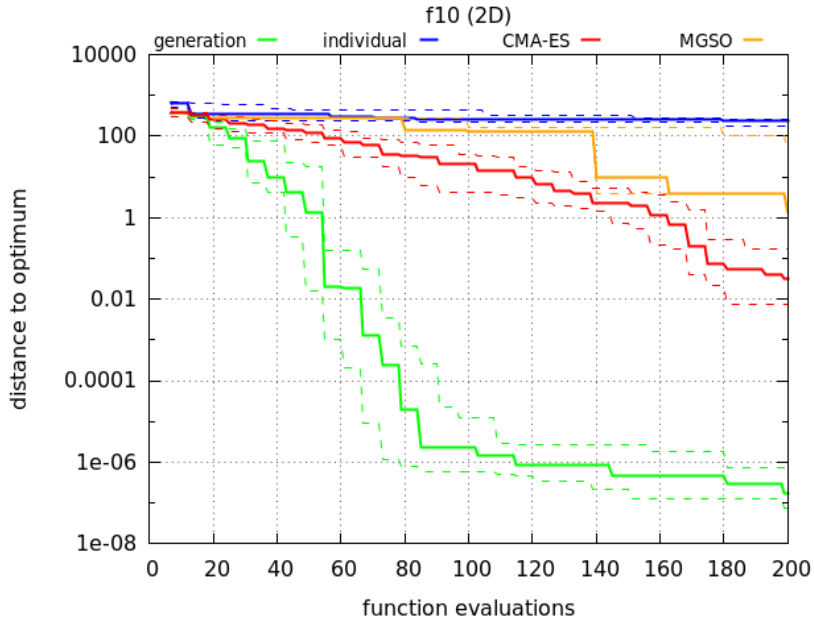

f11 (10D)
generation EC
individual EC
CMA-ES
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
100
200
300
400
500
600
700
800
900
1000

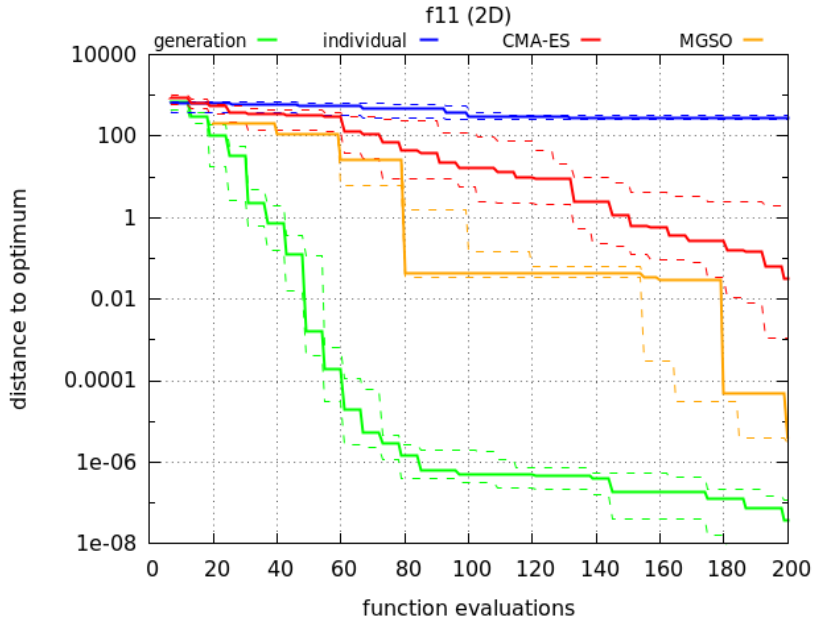
Part III

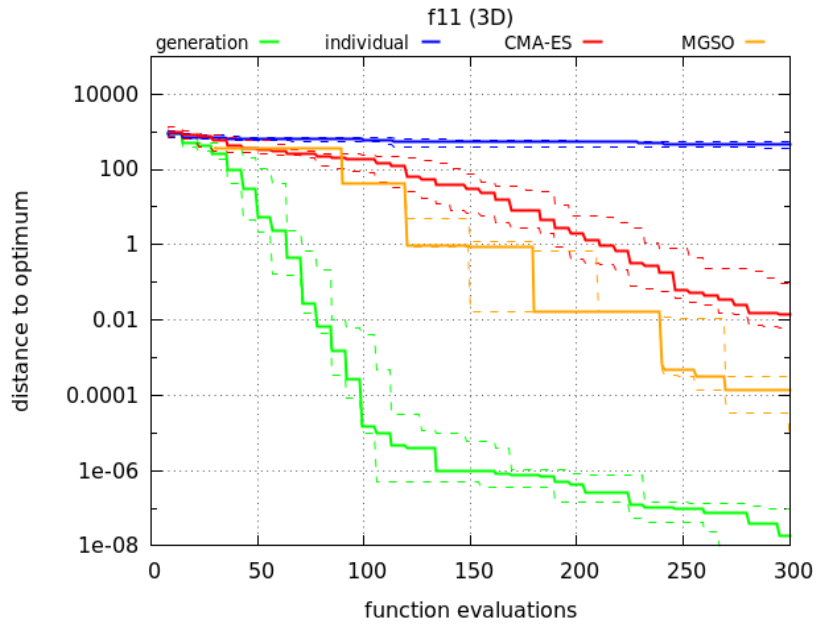
S-CMA-ES and MGSO comparison

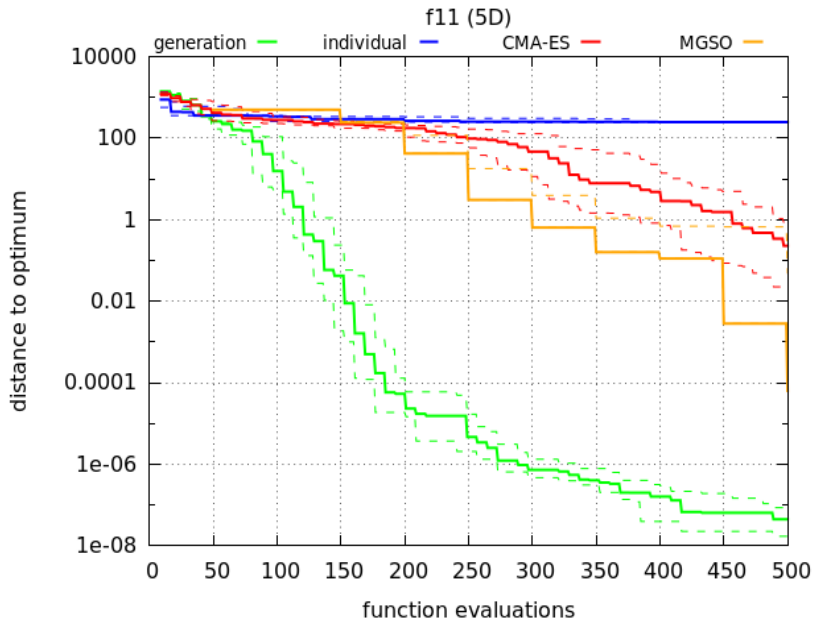

f8 (2D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200

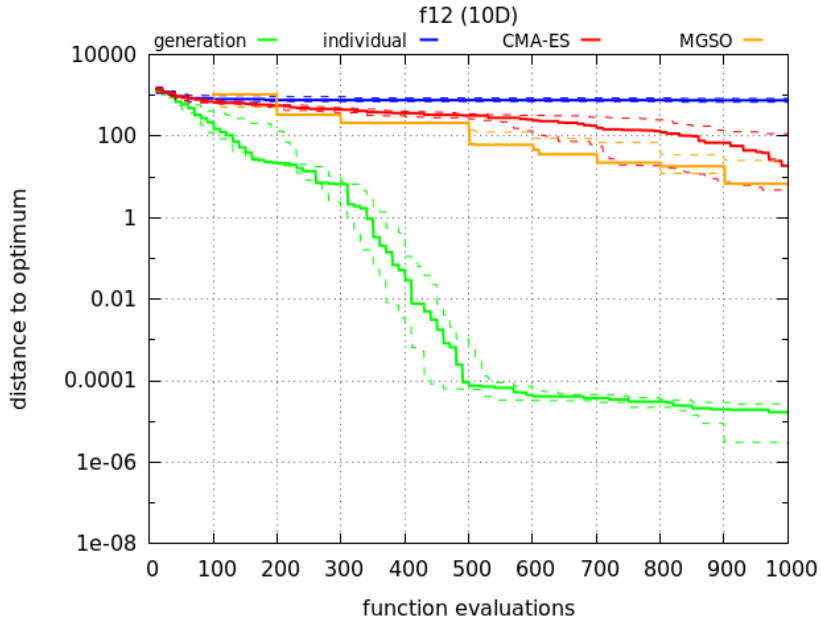

f9 (2D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200

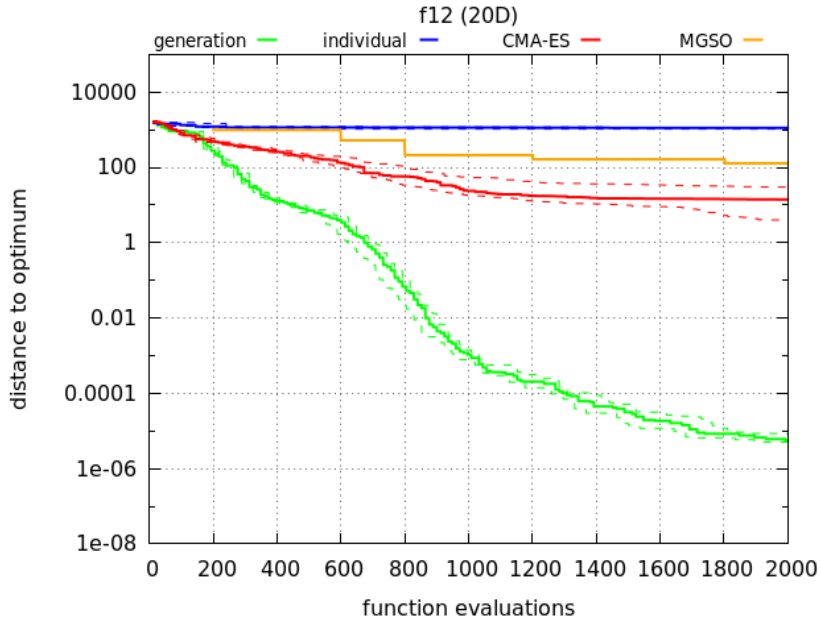

f10 (2D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f11 (2D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
20
40
60
80
100
120
140
160
180
200


f11 (3D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300


f11 (5D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
50
100
150
200
250
300
350
400
450
500


f12 (10D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
100
200
300
400
500
600
700
800
900
1000


f12 (20D)
generation
individual
CMA-ES
MGSO
distance to optimum
function evaluations
10000
100
1
0.01
0.0001
1e-06
1e-08
0
200
400
600
800
1000
1200
1400
1600
1800
2000