Analyzing black-box optimization algorithms - why and how?

Carola Doerr
CNRS research director @ Sorbonne Université, Paris, France

Journées ALEA, CIRM

supported by

Analyzing black-box optimization algorithms - why and how?

Carola Doerr
CNRS research director @ Sorbonne Université, Paris, France

Journées ALEA, CIRM

supported by

Analyzing black-box optimization algorithms - why and how?

Carola Doerr
CNRS research director @ Sorbonne Université, Paris, France

Journées ALEA, CIRM

supported by

Optimization (in its simplest form)

optimization: for a given function $f: S \to \mathbb{R}$, find $x \in S$ with $f(x)$ maximal/minimal

Examples:

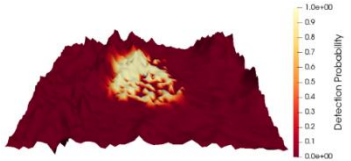
- TSP: traveling salesperson problem
- shortest path between two vertices in a graph
- grid percolation model: smallest radius such that the graph is connected
- …

(more or less)
well-studied (academic) problems,
often with hand-crafted solutions

Optimization in (my) real life

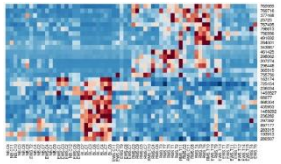
complex problems

logistics



network optimization

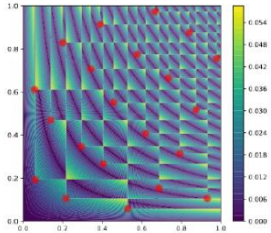
biomedicne



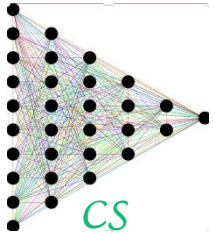
engineering

high time pressure

economics



math (!)



CS

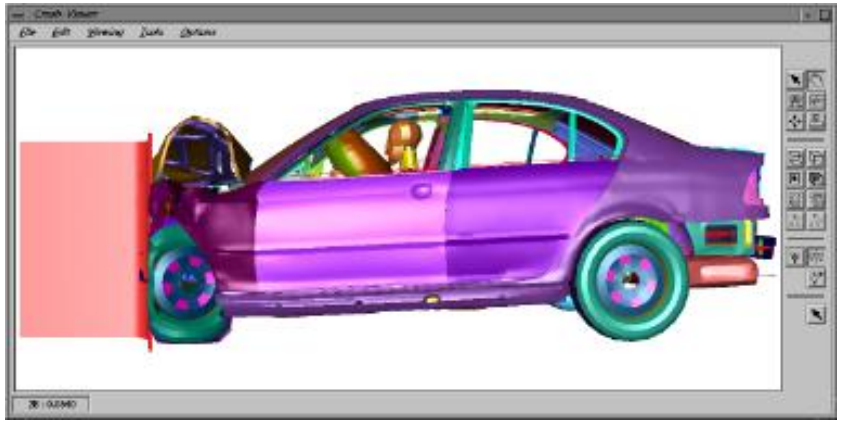
etc...

most pictures taken from https://pixabay.com/

Black-box optimization

Evaluations:

- simulations



[Picture: Dissertation Norbert Frisch]

Black-box optimization

Evaluations:

- simulations
- physical experiments

pictures taken from https://pixabay.com/

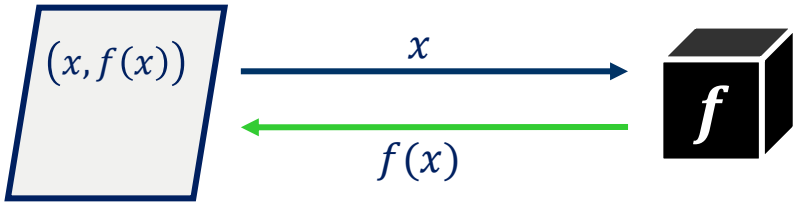
Black-box optimization

Evaluations:

- simulations
- physical experiments
- user study

Bouillabaisse, picture taken from https://www.tasteatlas.com/best-dishes-in-marseille

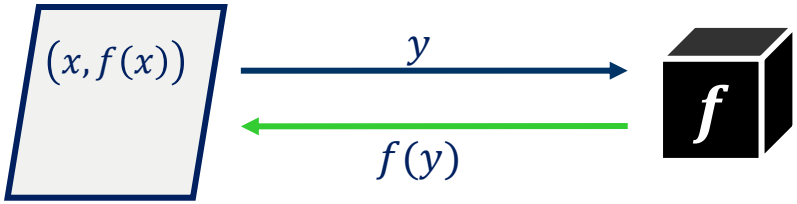
Black-box optimization



Evaluations:

- simulations
- physical experiments
- user study

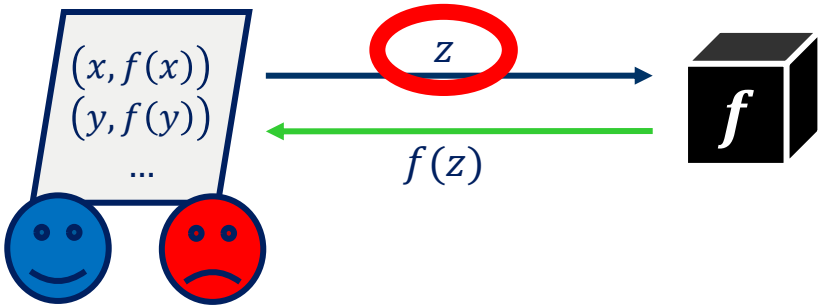
Black-box optimization



Evaluations:

- simulations
- physical experiments
- user study

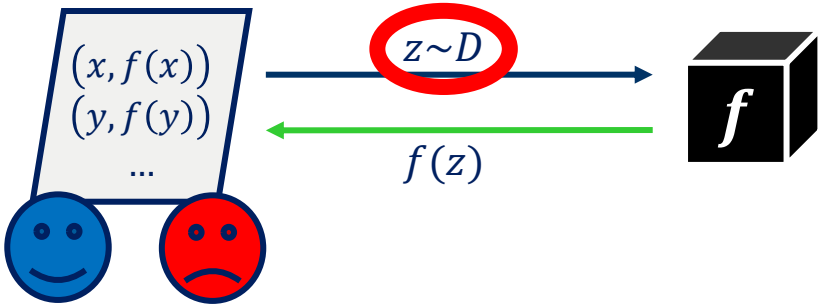
Black-box optimization



Evaluations:

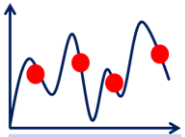
- simulations
- physical experiments
- user study

Black-box optimization

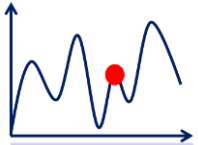


Evaluations:

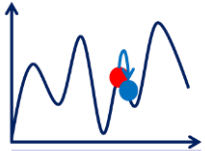
- simulations
- physical experiments
- user study



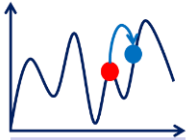
(quasi-, pseudo-) Random Sampling



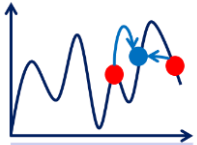
Local Search



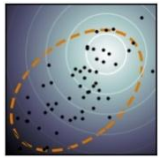
Simulated Annealing



Evolutionary Algorithms

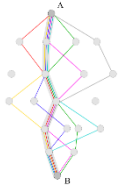


Genetic Algorithms

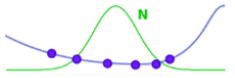


picture source: Wikipedia, 04/2021

Population-based Algo.

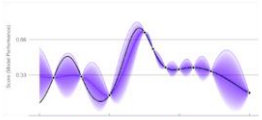


Swarm Intelligence



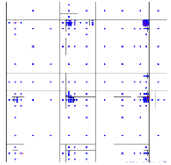
picture source: Wikipedia, 04/2021

Estimation of Distribution Algo.



picture source: Wikipedia, 04/2021

Surrogate-Based Opt. ("Bayesian Opt.")



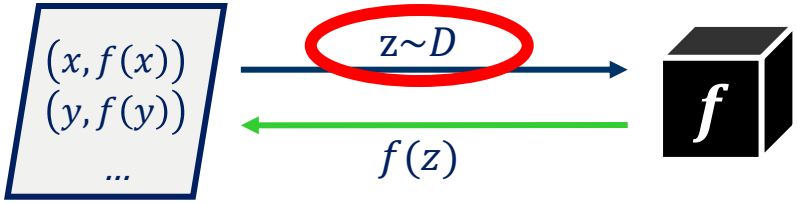
picture source: Munos AAAI 2013 tutorial

Partition-based Methods (Nelder-Mead, Powell,...)

many (MANY!) more...

Which algorithm to use for which problem?

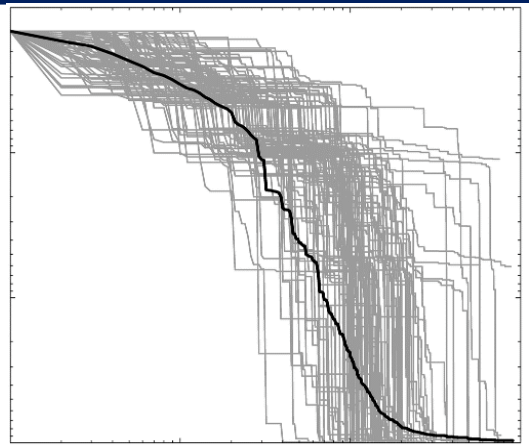
Complexity of analyzing black-box optimization algorithms



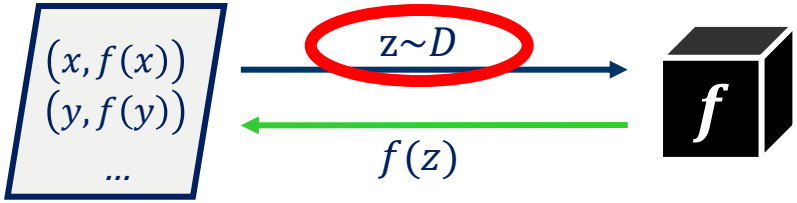
Evaluations:

- simulations
- physical experiments
- user study

stochastic behavior



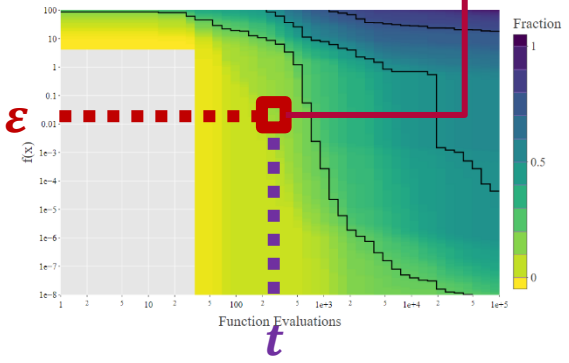
Complexity of analyzing black-box optimization algorithms



Evaluations:

- simulations
- physical experiments
- user study

stochastic behavior



attainment function:
probability that algo A finds a solution of loss $\leq \varepsilon$ within the first t evaluations

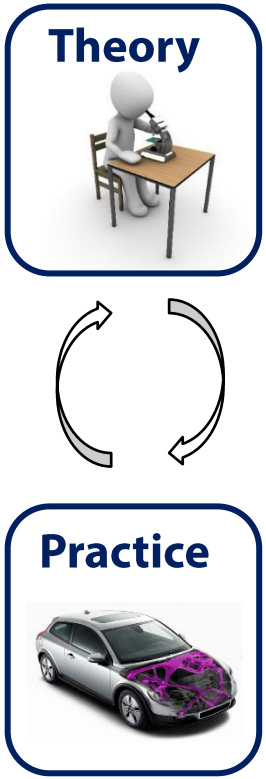
in discrete optimization: expected optimization time

(Some of) our research objectives

1. **mathematical curiosity:** understand the performance and the behavior of black-box optimization algorithms
   and not: given a problem, design an efficient algorithm

2. **configuration:** how does the performance depend on chosen (hyper-/control)parameters?

3. **selection:** which algorithm to use for what kind of problems?



Example

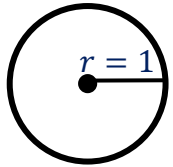
Randomized Local Search (RLS) for maximizing $f: \{0,1\}^n \to \mathbb{R}$

Initialization:

Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

Optimization: in iteration $t = 1,2,\ldots$ do

1. *variation:* create y from x by flipping a randomly chosen bit
2. *greedy selection:* if $f(y) \geq f(x)$, replace x by y



- ✓ simple
- ✓ no memory needed
- ? slow (?)
- × can get stuck in local optima

Example

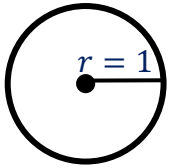
Randomized Local Search (RLS) for maximizing $f: \{0,1\}^n \to \mathbb{R}$

Initialization:

Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

Optimization: in iteration $t = 1,2,\ldots$ do

1. *variation:* create y from x by flipping a randomly chosen bit
2. *greedy selection:* if $f(y) \geq f(x)$, replace x by y

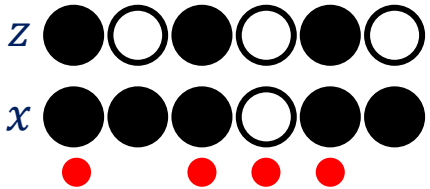


- ✓ simple
- ✓ no memory needed
- ? slow (?)
- × can get stuck in local optima

The OneMax problem *(Mastermind with 2 colors)*

Unknown target vector $z \in \{0,1\}^n$

Score of a search point x is $\#\{i \mid x_i = z_i\} = n - H(x,z)$



$f(x) = 4$

expected optimization time for RLS to solve OneMax?

when $f(x) = i$, then $\mathbb{P}[f(y) = i+1] = \frac{n-i}{n}$

coupon collector (with random initial assets) → $(1 \pm o(1))\, n \ln(n)$

Theorem (Erdös, Renyi 1963): Query complexity of OneMax is $\Theta(n/\log n)$

Proof: (upper bound) sample $2n/\log n$ strings u.a.r. (+some basic math).

Example

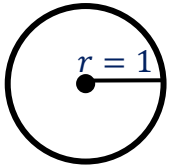
Randomized Local Search (RLS) for maximizing $f: \{0,1\}^n \to \mathbb{R}$

Initialization:

Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

Optimization: in iteration $t = 1,2,\ldots$ do

1. *variation:* create y from x by flipping a randomly chosen bit
2. *greedy selection:* if $f(y) \geq f(x)$, replace x by y

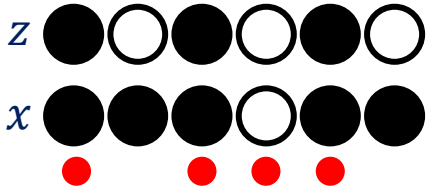


- ✓ simple
- ✓ no memory needed
- ? slow (?)
- × can get stuck in local optima

The OneMax problem *(Mastermind with 2 colors)*

Unknown target vector $z \in \{0,1\}^n$

Score of a search point x is $\#\{i \mid x_i = z_i\} = n - H(x,z)$



$f(x) = 4$

expected optimization time for RLS to solve OneMax?

when $f(x) = i$, then $\mathbb{P}[f(y) = i+1] = \frac{n-i}{n}$

coupon collector (with random initial assets) → $(1 \pm o(1))\, n \ln(n)$

Theorem (Erdös, Renyi 1963): Query complexity of OneMax is $\Theta(n/\log n)$

Proof: (upper bound) sample $2n/\log n$ strings u.a.r. (+some math). (lower bound) information theory

Example

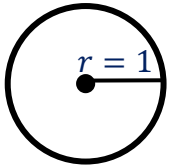
Randomized Local Search (RLS) for maximizing $f: \{0,1\}^n \to \mathbb{R}$

Initialization:

Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

Optimization: in iteration $t = 1,2,\ldots$ do

1. *variation:* create y from x by flipping a randomly chosen bit
2. *greedy selection:* if $f(y) \geq f(x)$, replace x by y



- ✓ simple
- ✓ no memory needed
- ? slow (?)
- × can get stuck in local optima

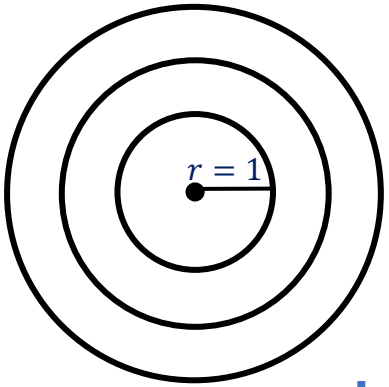
change neighborhood from which candidates are sampled

- active control of radius ("*step size*")
- random choice of radius ("*step size*")
- …

change selection to less greedy (e.g., Simulated Annealing)

…

Example for parameter control



Randomized Local Search (RLS)
with active ε-greedy multi-armed bandit parameter control

given: possible parameter values (=radii) $r_1, \dots, r_k$

Initialization:

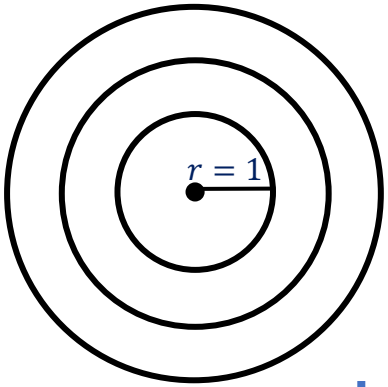
Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

Initialize all confidences $c_1, \dots, c_k$ as $1/k$

Optimization: in iteration $t = 1,2, \dots$ do

1. *choose radius:*
   - with probability *1-*ε, choose radius r with highest confidence (ties broken u.a.r.)
   - choose r u.a.r. otherwise
2. *variation:* create y from x by flipping r randomly chosen bit
3. *greedy selection:* if $f(y) \geq f(x)$, replace x by y
4. update the confidences $c_1, \dots, c_k$

Example for random parameter choice



The (1+1) Evolutionary Algorithm with mutation rate $0 < p < 1$
= Randomized Local Search (RLS) with randomly chosed radius

Initialization:

Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).

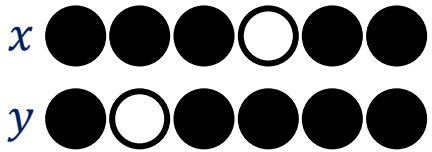
Optimization: in iteration $t = 1,2,\ldots$ do

1. *choose radius* r from binomial distribution $\mathrm{Bin}(n,p)$
2. *variation:* create y from x by flipping r randomly chosen bit
3. *greedy selection:* if $f(y) \geq f(x)$, replace x by y

Why "*evolutionary*" *algorithm*?

alternative way to do steps 1 and 2:

create y from x by flipping each bit with probability $1/n$



The (1+1) EA behaves like RLS "on average" but can escape local optima.

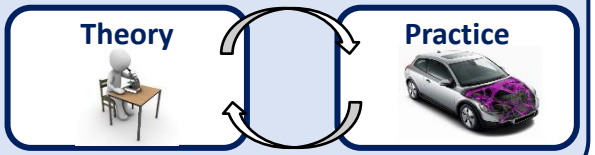
Expected optimization time on OneMax: $\leq en \ln n$

goals of the ERC project (in a nutshell)

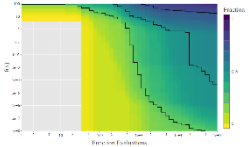
"from controlling parameters to dynamic choice of algorithms"

leverage strengths of different algorithms

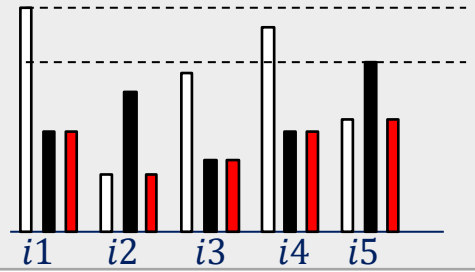
- ✓ for different phases of the optimization process
- ✓ for different types of problems (type not known in advance!)



Key challenges (theory):



- forces us to understand attainment functions
  --> much more fine-grained than "just" expected optimization times
- warm-starting: how to hand over information from one algo to another one?
- from worst-case analysis to instance-based analysis



Algorithm 1

Algorithm 1

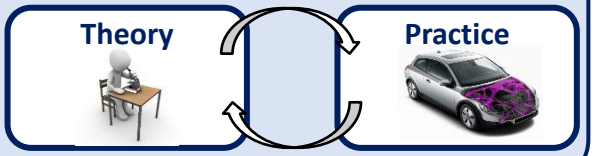
combined algorithm using instance-based selection

goals of the ERC project (in a nutshell)

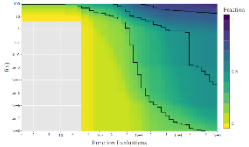
“from controlling parameters to dynamic choice of algorithms”

leverage strengths of different algorithms

- ✓ for different phases of the optimization process
- ✓ for different types of problems (type not known in advance!)



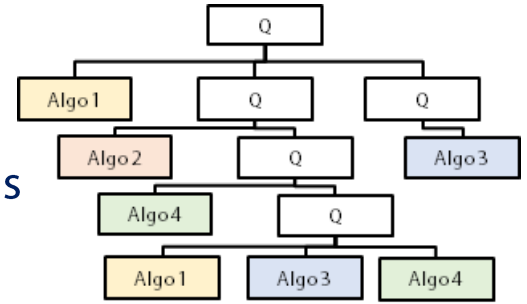
Key challenges (theory):



- forces us to understand attainment functions
  --> much more fine-grained than “just” expected optimization times
- warm-starting: how to hand over information from one algo to another one?
- from worst-case analysis to instance-based analysis

Key challenges (practice):

- develop guidelines for default solver(s) for popular settings

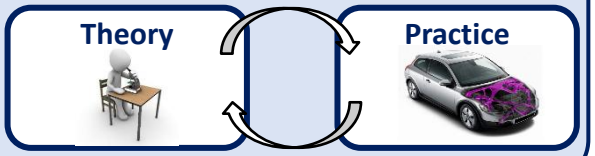


goals of the ERC project (in a nutshell)

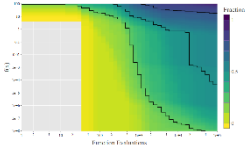
"from controlling parameters to dynamic choice of algorithms"

leverage strengths of different algorithms

- ✓ for different phases of the optimization process
- ✓ for different types of problems (type not known in advance!)



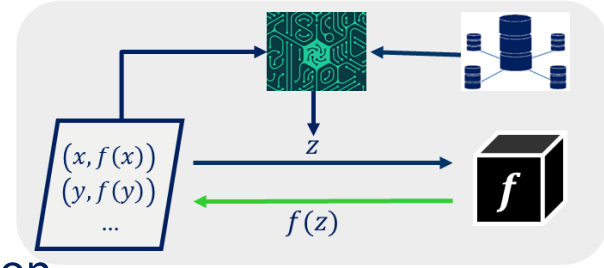
Key challenges (theory):



- forces us to understand attainment functions
  --> much more fine-grained than "just" expected optimization times
- warm-starting: how to hand over information from one algo to another one?
- from worst-case analysis to instance-based analysis

Key challenges (practice):

- develop guidelines for default solver(s) for popular settings
- use of machine learning techniques to guide dynamic selection



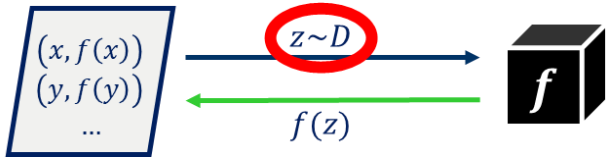
Take-home message(s)

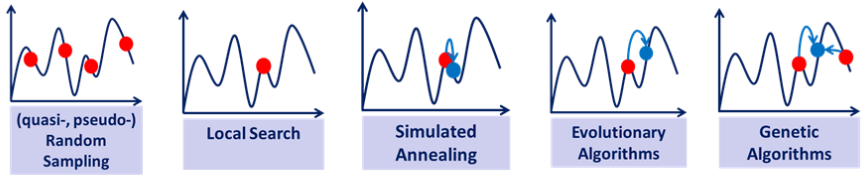
Black-box optimization algorithms are

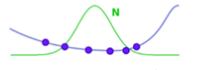
✓ quite crucial for solving many real-world problems

× not well understood (neither theoretically nor practically/empirically)

Many randomized decisions – we probably use many similar tools (concentration bounds, potential function arguments, drift theorems,…)

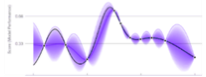






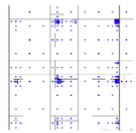
picture source: Wikipedia, 04/2021

Estimation of Distribution Algo.



picture source: Wikipedia, 04/2021

Surrogate-Based Opt. ("Bayesian Opt.")



picture source: Munos AAAI 2013 tutorial

Partition-based Methods (Nelder-Mead, Powell,...)

many (MANY!) more...