Theory of Discrete Black-Box Optimization:
what is it and why bother?

Carola Doerr

CNRS and Pierre et Marie Curie University, Paris, France

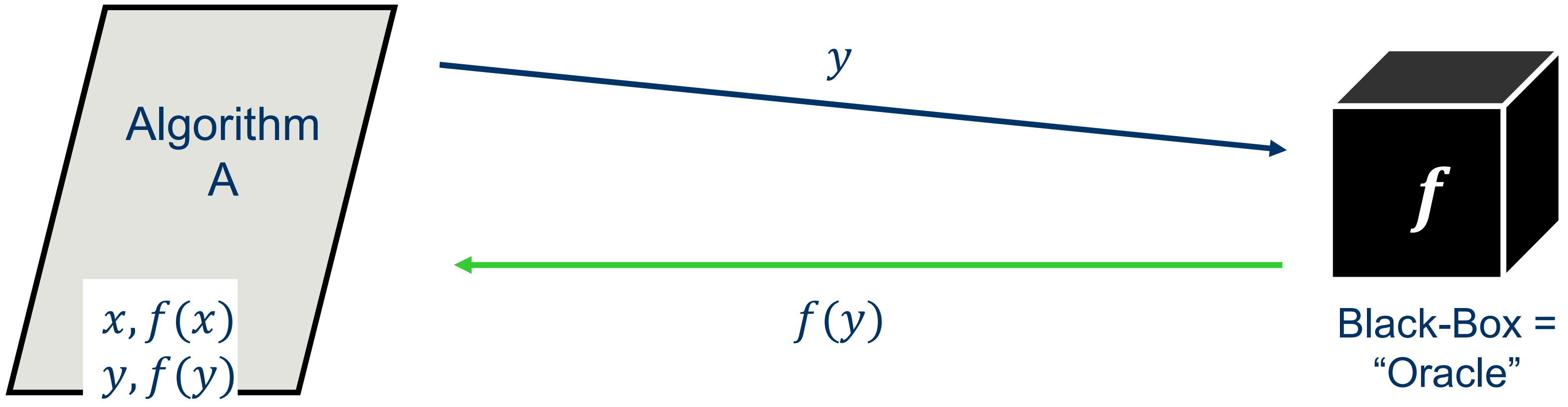
1st SIGEVO Summer School

Berlin, July 14, 2017

Black-Box Optimization

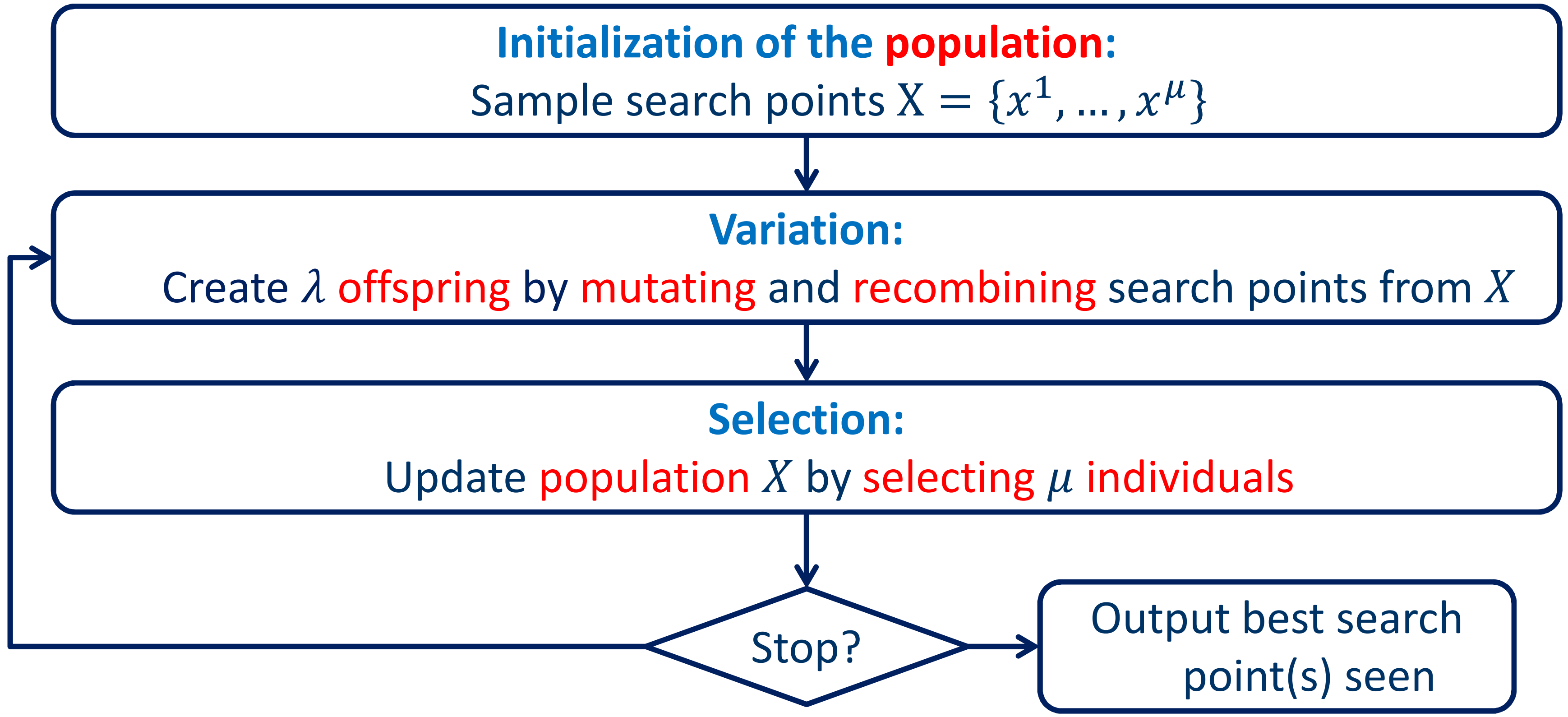
- Many real-world problems are essentially black-box problems

$f \in F \subseteq \{g: S \to \mathbb{R}\}$



- **Goal:** maximize f using as few function evaluations (aka *queries*) as possible
- **Main motivation:**
  - complexity of the problem → no solid investigation possible
  - privacy concerns → data owner does not want to disclose too much info
  - means to abstract away details (very successfully used in many state of the art solvers for hard combinatorial problems like MaxSAT, TSP, etc.)

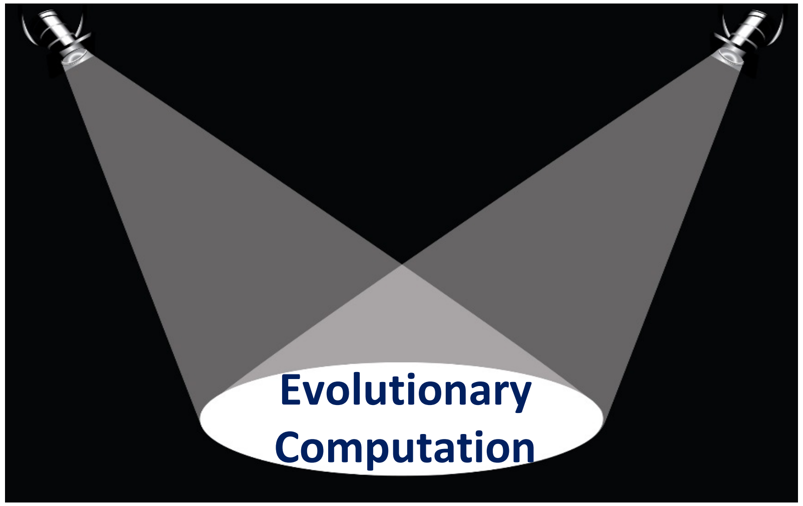
Evolutionary Computation as Black-Box Optimizers


Initialization of the population:
Sample search points $X = \{x^1, \dots, x^\mu\}$
Variation:
Create λ offspring by mutating and recombining search points from X
Selection:
Update population X by selecting μ individuals
Stop?
Output best search point(s) seen

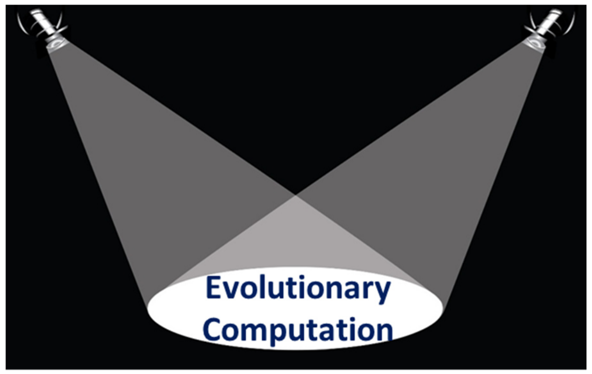
- For many real-world (industrial and academic) problems, EC approaches have proven quite successful:
  - they often give reasonable solutions in a reasonable amount of time
  - sometimes the only means to find any reasonable solution at all

Research in EC (1/2)

- **Research in EC** and related fields (optimization, operations research, artificial intelligence, machine learning, …):
  - when and why (!) do EAs (and similar heuristics) work well?
  - how to optimally tune your algorithm to the problem at hand?
- Several different approaches to address these questions (see variety of GECCO talks…). Essentially, they can be categorized as
  - *empirical research*, aiming to gain insights by means of well-designed experiments
  - *theoretical research*, aiming to gain insights by means of mathematical precision

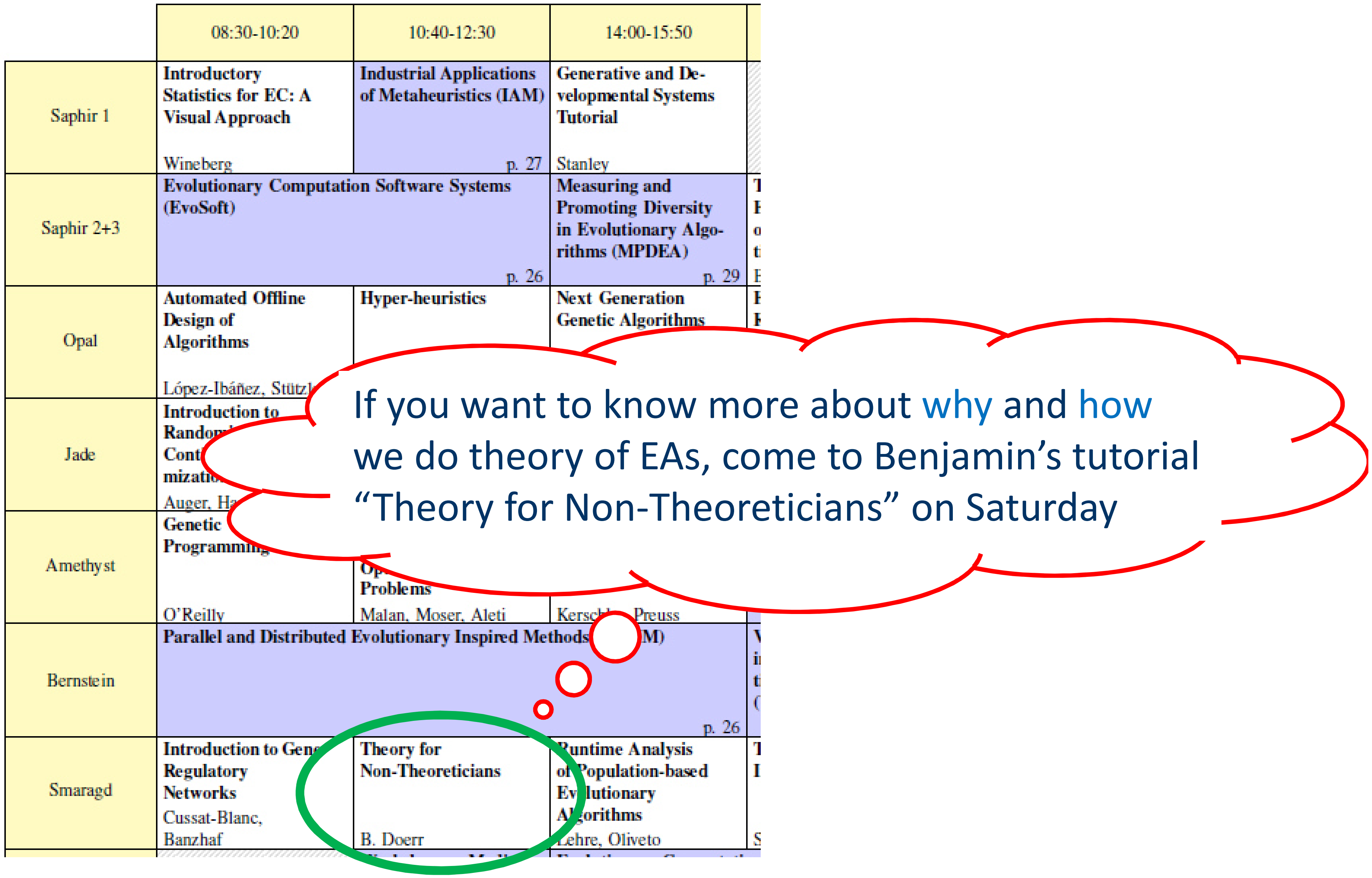


Research in EC (2/2)



- Interestingly (and unfortunately !), these 2 approaches are often viewed as independent or even conflicting
- **Goal of my talk:** To convince you that this view is not only wrong, it can also be quite harmful to our community
- **My proposition:** view empirical and theoretical research as complementary research methods, (i.e., combining these 2 views, we get a much more complete picture of strengths and weaknesses of EC approaches)

What Do We Mean With *Theory of EC* ?



	08:30-10:20	10:40-12:30	14:00-15:50
Saphir 1	**Introductory Statistics for EC: A Visual Approach** Wineberg	**Industrial Applications of Metaheuristics (IAM)** p. 27	**Generative and Developmental Systems Tutorial** Stanley
Saphir 2+3	**Evolutionary Computation Software Systems (EvoSoft)** p. 26		**Measuring and Promoting Diversity in Evolutionary Algorithms (MPDEA)** p. 29
Opal	**Automated Offline Design of Algorithms** López-Ibáñez, Stütz	**Hyper-heuristics**	**Next Generation Genetic Algorithms**
Jade	**Introduction to Random Cont mizatio** Auger, Ha		
Amethyst	**Genetic Programming** O'Reilly	**Op Problems** Malan, Moser, Aleti	Kersch Preuss
Bernstein	**Parallel and Distributed Evolutionary Inspired Methods M)** p. 26		
Smaragd	**Introduction to Gene Regulatory Networks** Cussat-Blanc, Banzhaf	**Theory for Non-Theoreticians** B. Doerr	**Runtime Analysis of Population-based Evolutionary Algorithms** Lehre, Oliveto

If you want to know more about why and how we do theory of EAs, come to Benjamin's tutorial "Theory for Non-Theoreticians" on Saturday

What Do We Mean With *Theory of EC* ?

- Results proven with mathematical rigor
- Mathematical rigor:
  - make precise the *evolutionary algorithm* (EA) you regard
  - make precise the *problem* you try to solve with the EA
  - make precise a statement on the *performance* of the EA solving this problem
  - *prove* this statement by mathematical means

Theoretical Results - Example

should be made precise in the paper to avoid any ambiguity

(1+1) evolutionary algorithm to maximize $f: \{0,1\}^n \to \mathbb{R}$:

```
1. choose $x \in \{0,1\}^n$ uniformly at random
2. while not *terminate* do
3.     generate $y$ from $x$ by flipping each bit independently
       with probability $1/n$ (“standard-bit mutation”)
4.     if $f(y) \geq f(x)$ then $x := y$
5. output $x$
```

A mathematically proven result

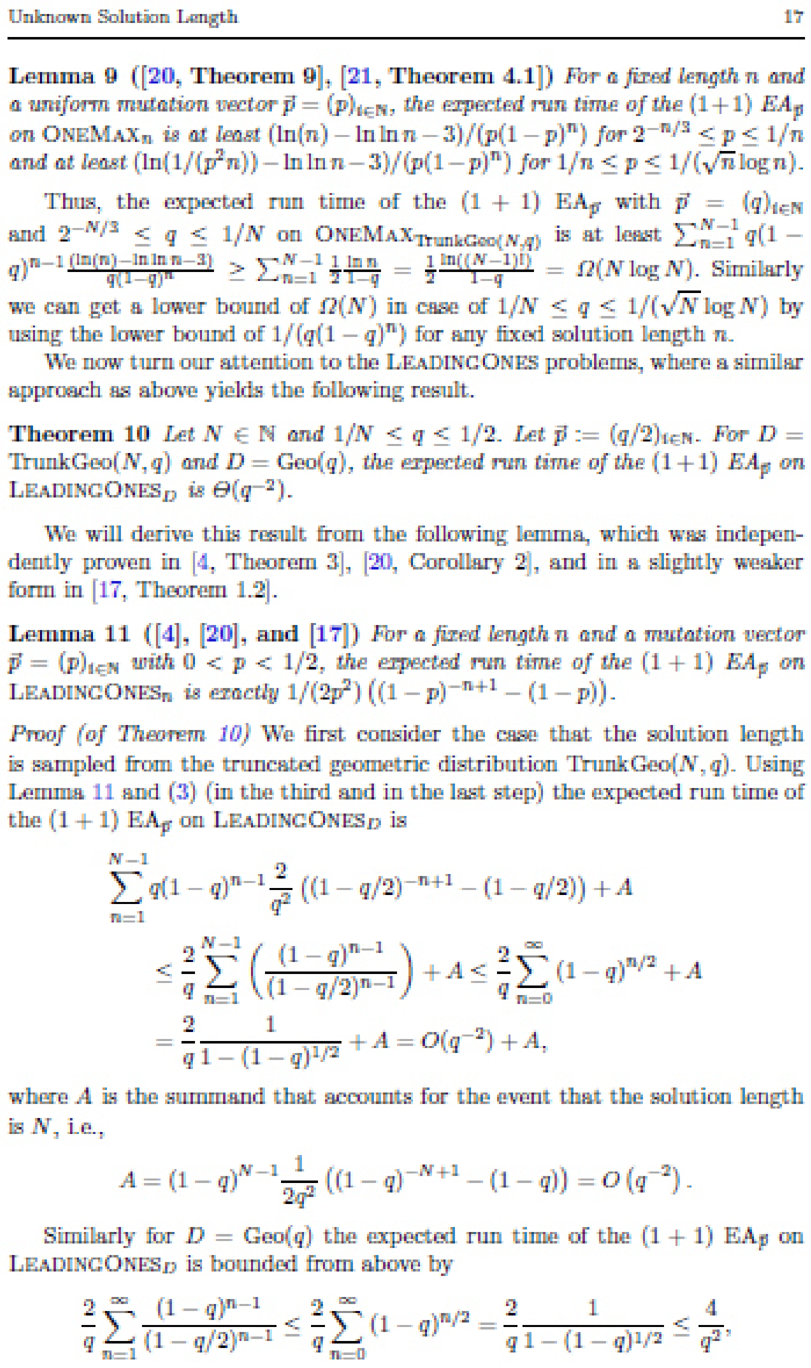
Theorem: The (1+1) evolutionary algorithm finds the maximum of any linear function

$$f: \{0,1\}^n \to \mathbb{R}, (x_1, \dots, x_n) \mapsto \sum_{i=1}^{n} w_i x_i, \qquad w_1, \dots, w_n \in \mathbb{R},$$

in an expected number of $O(n \log n)$ iterations.

at most $Cn \ln n$ for some unspecified constant C

performance measure: number of iterations or fitness evaluations, but not runtime in seconds



Lemma 9 **([20, Theorem 9], [21, Theorem 4.1])** *For a fixed length n and a uniform mutation vector $\vec{p} = (p)_{i \in \mathbb{N}}$, the expected run time of the* $(1+1)$ *EA*$_{\vec{p}}$ *on* OneMax$_n$ *is at least* $(\ln(n) - \ln\ln n - 3)/(p(1-p)^n)$ *for* $2^{-n/3} \le p \le 1/n$ *and at least* $(\ln(1/(p^2 n)) - \ln\ln n - 3)/(p(1-p)^n)$ *for* $1/n \le p \le 1/(\sqrt{n}\log n)$.

Thus, the expected run time of the $(1+1)$ EA$_{\vec{p}}$ with $\vec{p} = (q)_{i \in \mathbb{N}}$ and $2^{-N/3} \le q \le 1/N$ on OneMax$_{\text{TrunkGeo}(N,q)}$ is at least $\sum_{n=1}^{N-1} q(1-q)^{n-1} \frac{(\ln(n) - \ln\ln n - 3)}{q(1-q)^n} \ge \sum_{n=1}^{N-1} \frac{1}{2} \frac{\ln n}{1-q} = \frac{1}{2} \frac{\ln((N-1)!)}{1-q} = \Omega(N \log N)$. Similarly we can get a lower bound of $\Omega(N)$ in case of $1/N \le q \le 1/(\sqrt{N}\log N)$ by using the lower bound of $1/(q(1-q)^n)$ for any fixed solution length n.

We now turn our attention to the LeadingOnes problems, where a similar approach as above yields the following result.

Theorem 10 *Let $N \in \mathbb{N}$ and $1/N \le q \le 1/2$. Let $\vec{p} := (q/2)_{i \in \mathbb{N}}$. For $D =$ TrunkGeo(N,q) and $D =$ Geo(q), the expected run time of the* $(1+1)$ *EA*$_{\vec{p}}$ *on* LeadingOnes$_D$ *is* $\Theta(q^{-2})$.

We will derive this result from the following lemma, which was independently proven in [4, Theorem 3], [20, Corollary 2], and in a slightly weaker form in [17, Theorem 1.2].

Lemma 11 **([4], [20], and [17])** *For a fixed length n and a mutation vector $\vec{p} = (p)_{i \in \mathbb{N}}$ with $0 < p < 1/2$, the expected run time of the* $(1+1)$ *EA*$_{\vec{p}}$ *on* LeadingOnes$_n$ *is exactly* $1/(2p^2)\left((1-p)^{-n+1} - (1-p)\right)$.

Proof (of Theorem 10) We first consider the case that the solution length is sampled from the truncated geometric distribution TrunkGeo(N,q). Using Lemma 11 and (3) (in the third and in the last step) the expected run time of the $(1+1)$ EA$_{\vec{p}}$ on LeadingOnes$_D$ is

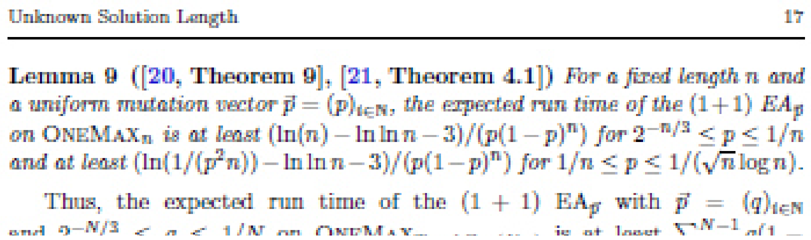
$$\begin{aligned}
&\sum_{n=1}^{N-1} q(1-q)^{n-1} \frac{2}{q^2}\left((1-q/2)^{-n+1} - (1-q/2)\right) + A \\
&\le \frac{2}{q} \sum_{n=1}^{N-1} \left(\frac{(1-q)^{n-1}}{(1-q/2)^{n-1}}\right) + A \le \frac{2}{q} \sum_{n=0}^{\infty} (1-q)^{n/2} + A \\
&= \frac{2}{q} \frac{1}{1-(1-q)^{1/2}} + A = O(q^{-2}) + A,
\end{aligned}$$

where A is the summand that accounts for the event that the solution length is N, i.e.,

$$A = (1-q)^{N-1} \frac{1}{2q^2}\left((1-q)^{-N+1} - (1-q)\right) = O\left(q^{-2}\right).$$

Similarly for $D =$ Geo(q) the expected run time of the $(1+1)$ EA$_{\vec{p}}$ on LeadingOnes$_D$ is bounded from above by

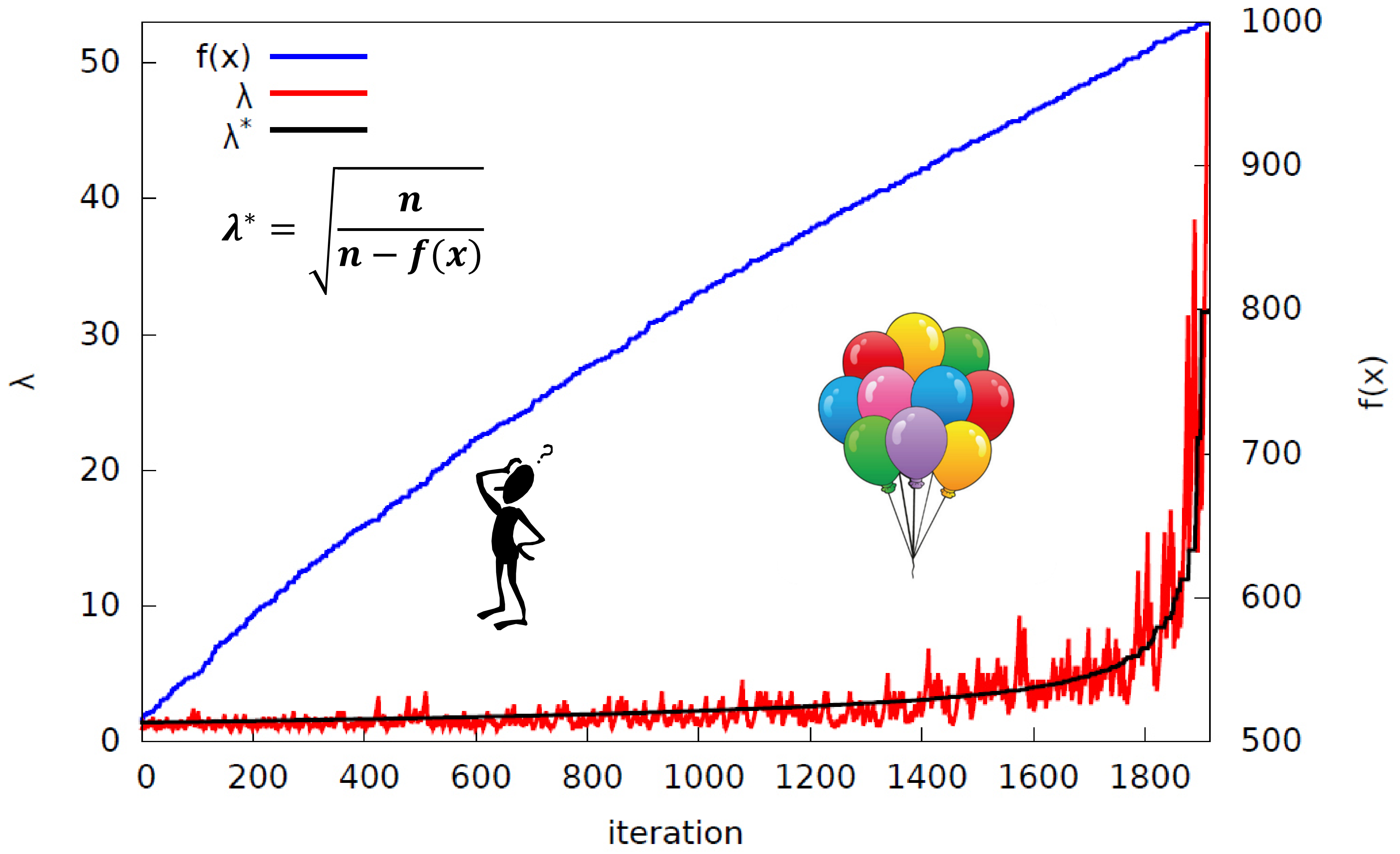
$$\frac{2}{q} \sum_{n=1}^{\infty} \frac{(1-q)^{n-1}}{(1-q/2)^{n-1}} \le \frac{2}{q} \sum_{n=0}^{\infty} (1-q)^{n/2} = \frac{2}{q} \frac{1}{1-(1-q)^{1/2}} \le \frac{4}{q^2},$$



Unknown Solution Length 17

Lemma 9 ([20, Theorem 9], [21, Theorem 4.1]) *For a fixed length n and a uniform mutation vector $\vec{p} = (p)_{i\in\mathbb{N}}$, the expected run time of the* (1+1) *EA$_{\vec{p}}$ on* ONEMAX$_n$ *is at least* $(\ln(n) - \ln\ln n - 3)/(p(1-p)^n)$ *for* $2^{-n/3} \le p \le 1/n$ *and at least* $(\ln(1/(p^2 n)) - \ln\ln n - 3)/(p(1-p)^n)$ *for* $1/n \le p \le 1/(\sqrt{n}\log n)$.

Thus, the expected run time of the (1 + 1) EA$_{\vec{p}}$ with $\vec{p} = (q)_{i\in\mathbb{N}}$ and $2^{-N/3} \le q \le 1/N$ on ONEMAX ... is at least $\sum^{N-1}$ $q(1 -$



$q \sum_{n=1} (1 - q/2)^{n-1} = q \sum_{n=0}$ ... $q\, 1 - (1-q)^{1/2} = q^2$

Why Do Theory? Because of Results

- Strong statements: Absolute guarantee that the result is correct

- Some results can only be obtained by theory; e.g., because you make a statement on a very large or even infinite set
  - all bit-strings of length n,
  - all TSP instances on n vertices,
  - all input sizes $n \in \mathbb{N}$,
  - all possible algorithms for a problem

- Mathematical analysis helps to identify the working principles of EAs
  $\rightarrow$ we will discuss in this talk how this can give surprising insights

- Theory can be fun!

Restrictions to Mathematical Approach: The “Theory Dilemma”

The high precision comes a certain a price…

Possible drawbacks of theory results include:

- Restricted scope: So far, mostly simple algorithms could be analyzed for simple optimization problems
- Less precise results: Constants are not tight, or not explicit as in “$O(n^2)$” = “less than cn^2 for some unspecified constant c”
- Less specific results: You get a weaker guarantee for all problem instances instead of a stronger one for the instances that show up in your real-world application
- Theory results can be very difficult to obtain: The proof might be short and easy to read, but finding it took long hours

In short: For problems allowing a mathematical analysis we can almost always design algorithms that are far better than any EC approaches. The power of EC is in *problems that we do not understand* with mathematical precision

Theory and Experiments: *Complementary Results*

Theory	Empirics
cover all problem instances of arbitrary sizes → guarantee!	only a finite number of instances of bounded size → have to see how representative this is
proof tells you the reason	only tells you numbers
only models for real-world instances (realistic?)	real-world instances
limited scope, e.g., (1+1) EA	everything you can implement
limited precision, e.g., $O(n^2)$	exact numbers
implementation independent	depends on implementation
finding proofs can be difficult	can be cheap (well, depends…)

→ Ideal: Combine theory and experiments. Difficulty: Get theoretically and empirically working researchers talk to each other…

Theory and Experiments: *Complementary Results*

THEORY

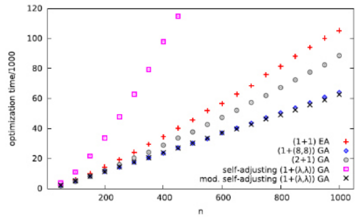
- cover all problem instances of arbitrary sizes
  → guarantee!
- proof tells you the
- only models for re
  instances (realisti
- limited scope, e.g
- limited precision, e
- implementation in
- finding proofs can be difficult

EXPERIMENTS

- only a finite number of instances of bounded size
  → have to see how
  resentative this is
- ells you numbers
- vorld instances
- thing you can implement
- numbers
- nds on implementation
- can be cheap (well, depends…)

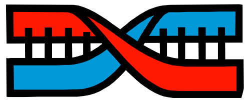
→ Ideal: Combine theory and experiments. Difficulty: Get theoretically and empirically working researchers talk to each other…

4 Selected Topic from Theory of EC



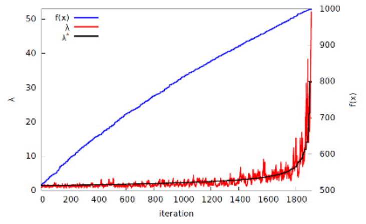
A. Runtime Analysis

B. Black-Box Complexity



C. Role of Crossover

D. Self-Adjusting Parameter Choices



Runtime Analysis

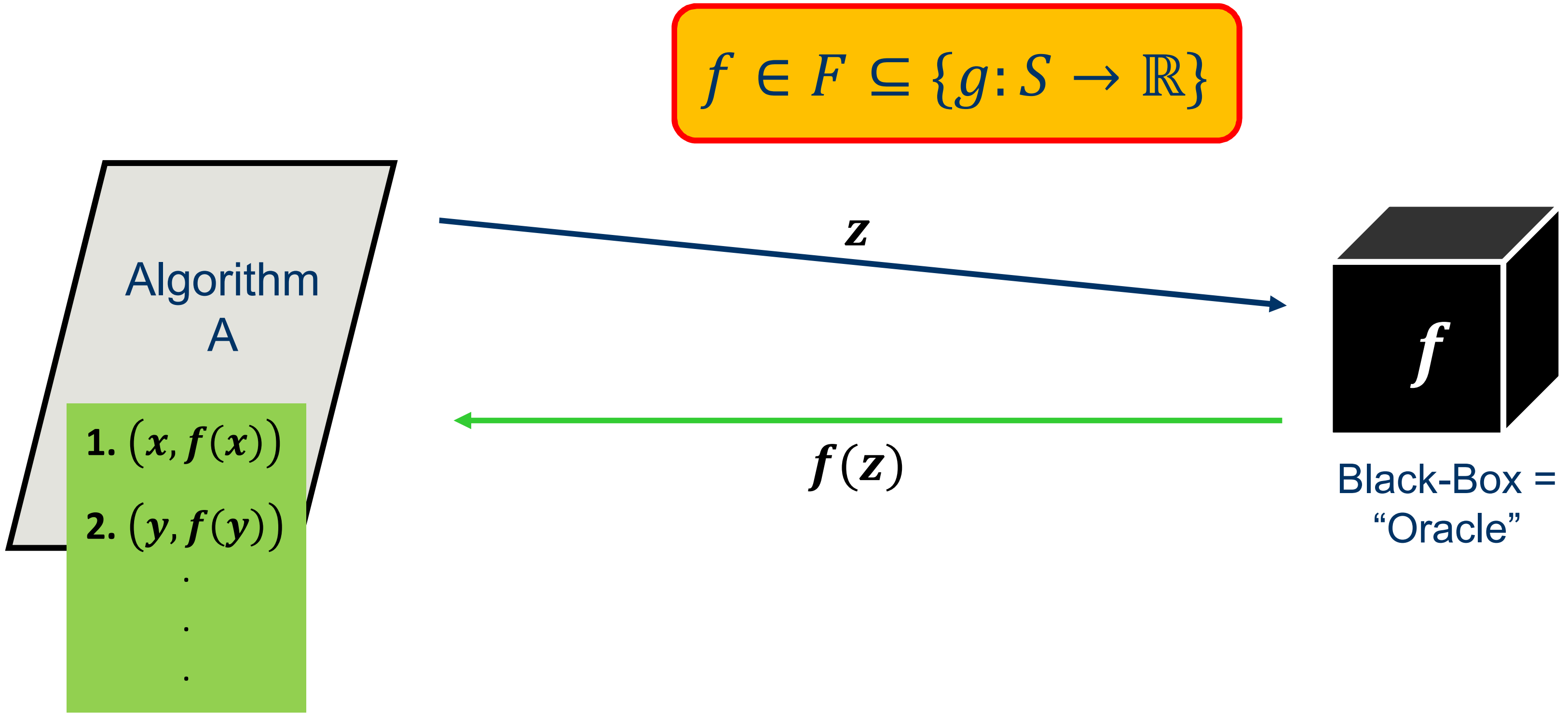
- Runtime analysis is *the* predominant topic in theory of EAs
- Runtime = optimization time = time needed until an optimal solution is queried for the first time
  (Note: "time" is typically measured by the number of function evaluations or generations)
- Aims at understanding how the performance of an algorithm depends
  - on the problem dimension (e.g., "the expected runtime of the (1+1) EA on any linear function is $\Theta(n \log n)$.")
  - on the different parameters involved (e.g., "the expected runtime of the (1+λ) EA with mutation rate $p = c/n$ on OneMax is XXX [some value that depends on λ, c and n].")
- Mathematically challenging already for quite simple EAs
  → has caught the attention of many researchers who do not necessarily care about biological background
  → *different motivations to study EAs exist*! (and this is reflected in the different results of very different flavor)

What Runtime Analysis Can Offer

1. Traditionally, main task of runtime analysis was probably to confirm and to debunk common beliefs/misconceptions that existed in the EC community
   - Are all functions w/o local optima are easily optimized by EAs?
   - Are all monotone functions easily optimized by EAs?
   - How important is crossover?
   - …
2. Today: turning point where runtime analysis starts to be helpful in choosing the right parameters, representations, operators, and algorithms (made possible by great progress in our analytical tools)
3. Recent theory has also inspired the design of new representations, operators, and algorithms

2nd Topic: Black-Box Complexity

- EAs are black-box optimizers



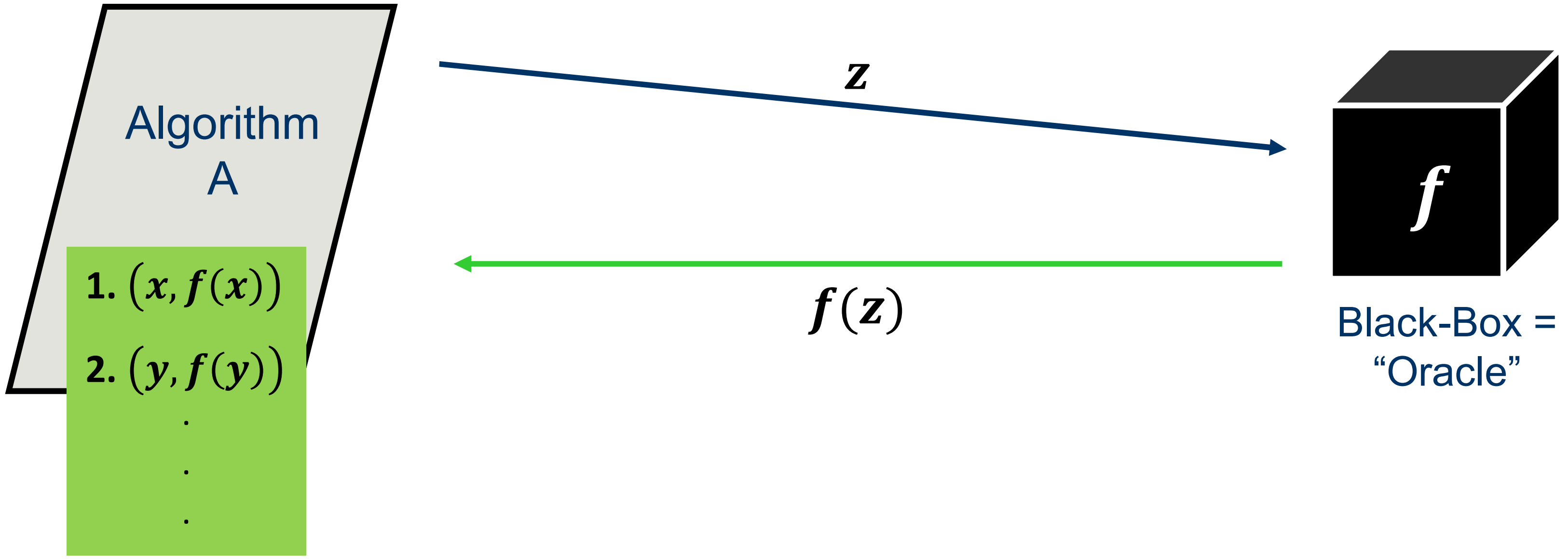
- Goal: maximize f (i.e., find a search point x such that $f(x) = \max\{f(s) \mid s \in S\}$)
- Performance measure (informally and only for discrete setting):

  Expected # of queries needed until for every $f \in F$ an optimal search point is evaluated for the first time

2nd Topic: Black-Box Complexity

- EAs are black-box optimizers

$f \in F \subseteq \{g: S \to \mathbb{R}\}$

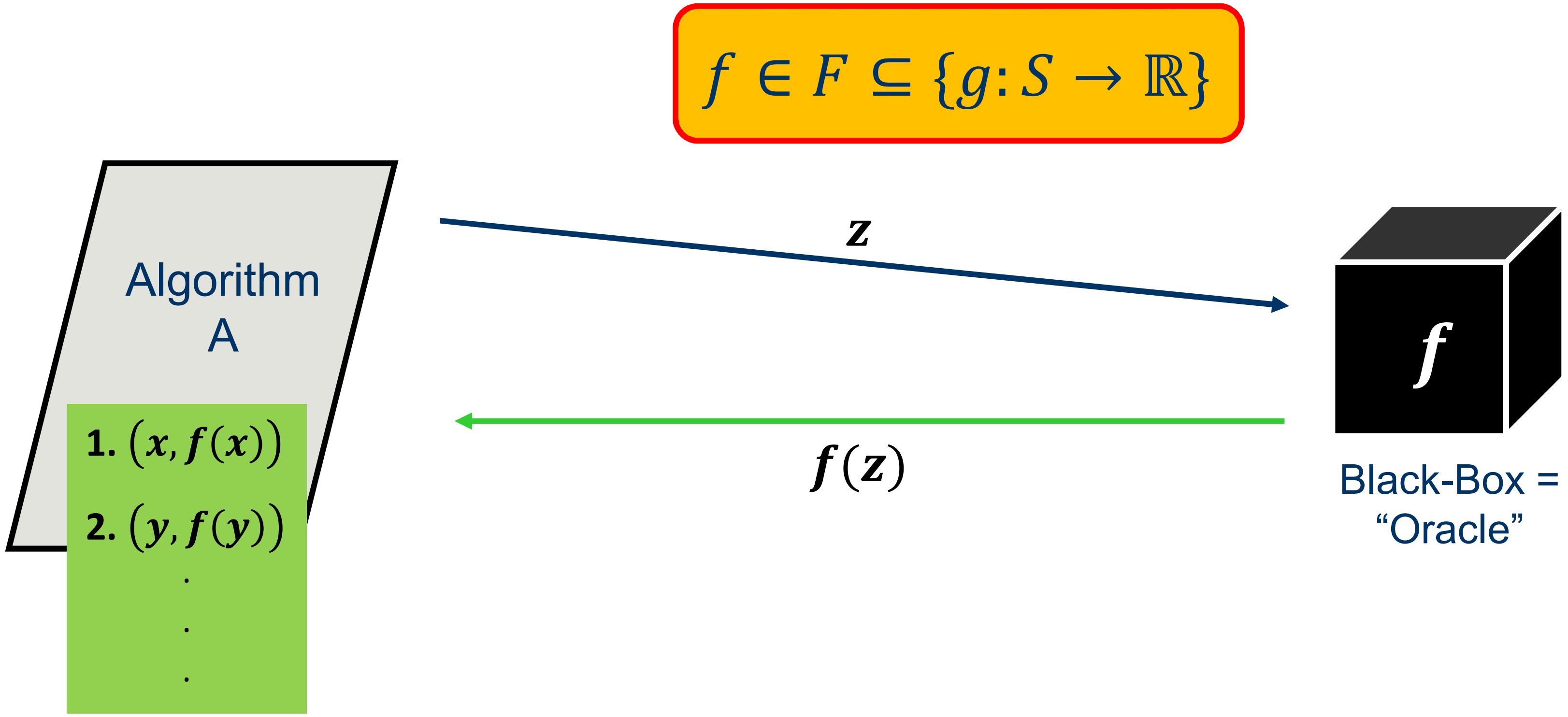


- **Black-Box Complexity:**

  Minimal expected # of queries needed until for every $f \in F$ an optimal search point is evaluated for the first time
- Informally: "*performance of the best possible algorithm*"

2nd Topic: Black-Box Complexity

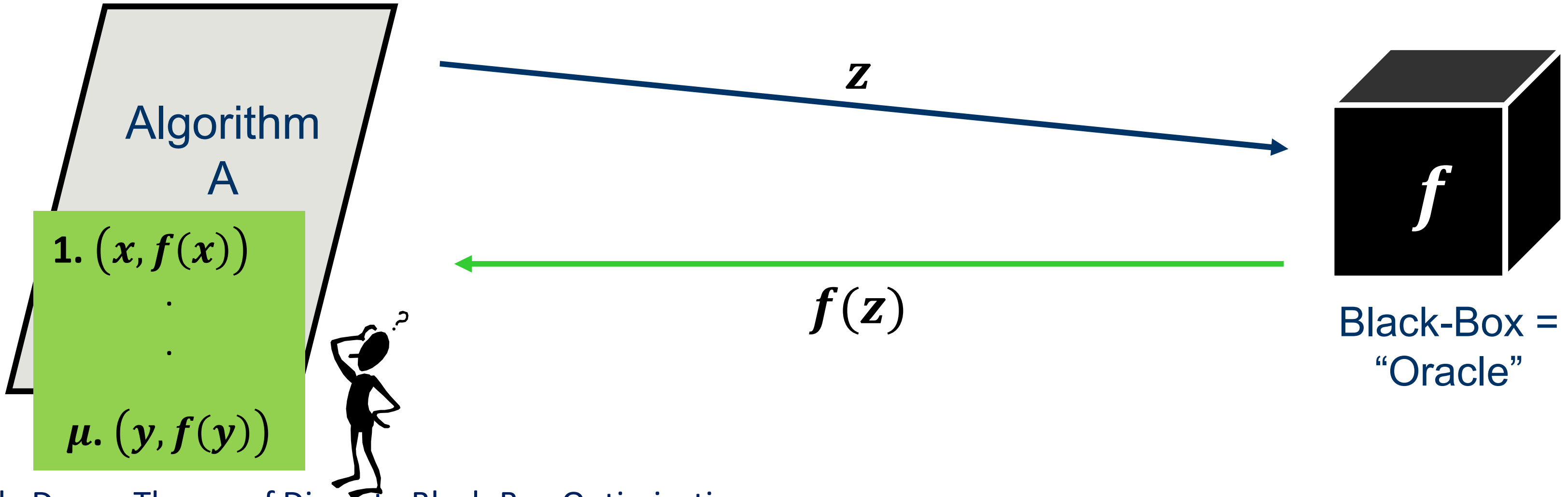
- EAs are black-box optimizers



- **Black-Box Complexity:**
  Minimal expected # of queries needed until for every $f \in F$ an optimal search point is evaluated for the first time
- Very strong results: *universal* lower bounds!

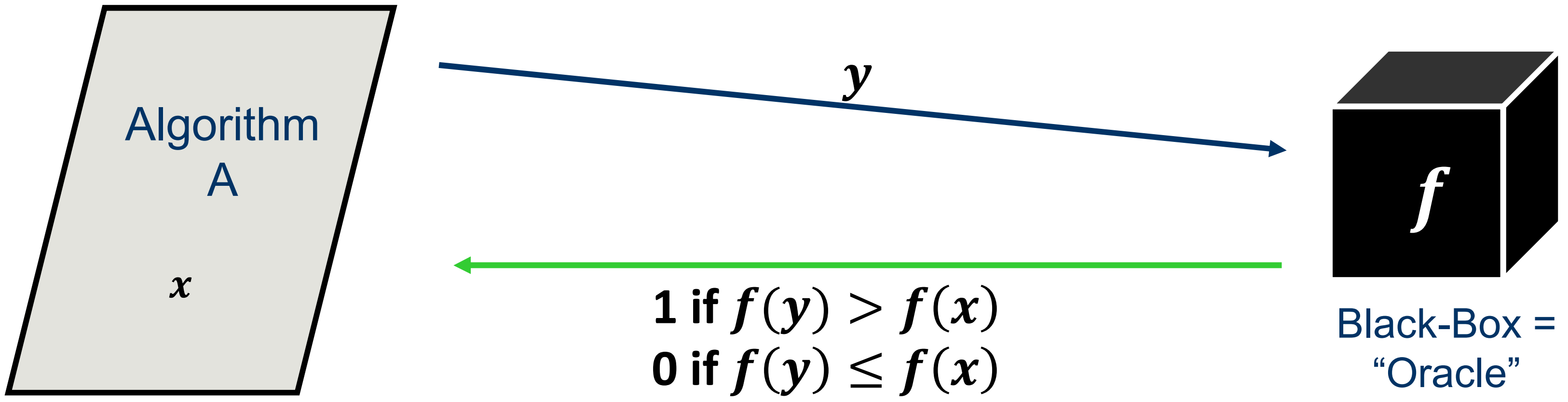
Restricted Black-Box Complexity Models

- Black-box complexities can be very low: universal model includes algorithms that are far from being an EA.
  Example: When $\mathrm{F} = \{f\}$, then the black-box complexity is only 1: the algorithm querying opt f in the first step is optimal!
- It is sometimes useful to regard larger classes of similar problems, e.g., $F = \{ g \mid g \text{ "similar to" } f \}$
- Alternatively, or in addition, one can also restrict the set of algorithms regarded:
  - Memory-restricted algorithms:



Restricted Black-Box Complexity Models

- Black-box complexities can be very low: universal model includes algorithms that are far from being an EA.
  Example: When $\mathrm{F} = \{f\}$, then the black-box complexity is only 1: the algorithm querying opt f in the first step is optimal!
- It is sometimes useful to either regard larger classes of similar problems, e.g., $F = \{\, g \mid g \text{ "similar to" } f \,\}$
- Alternatively, or in addition, one can also restrict the set of algorithms regarded:
  - (1+1) comparison-based algorithms:

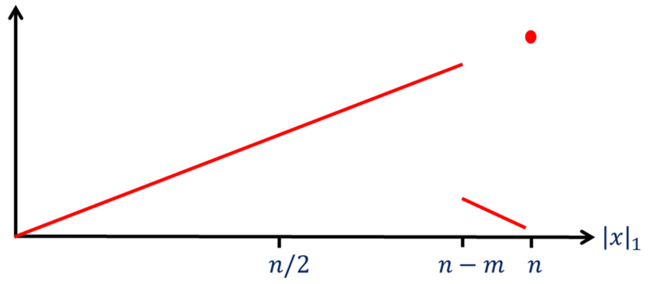


Restricted Black-Box Complexity Models

- Black-box complexities can be very low: universal model includes algorithms that are far from being an EA.
  Example: When $\mathrm{F} = \{f\}$, then the black-box complexity is only 1: the algorithm querying opt f in the first step is optimal!
- It is sometimes useful to either regard larger classes of similar problems, e.g., $F = \{ g \mid g \text{ "similar to" } f \}$
- Alternatively, or in addition, one can also restrict the set of algorithms regarded:
  - Unbiased algorithms [LehreWitt12]: algorithms that are translation- and transposition-invariant. For $S = \{0,1\}^n$ they treat bit positions and bit values identically
    - *unary* unbiased: basically means mutation-only algorithms. All search points are sampled uniformly or based on one previous sample
    - *binary* unbiased: combinations of 2 search points possible, but operator should not discriminate between positions and values
    - 3-ary, 4-ary, etc

3rd Topic: Usefulness of Crossover (1/2)

- One of the long-standing open problems in EC:
  Usefulness of Crossover in an Optimization Context
- Several attempts to prove usefulness of X-over:
  - Jansen, Wegener:
    - Jump function
    - showed that uniform crossover can help
    - proof requires a very small crossover rate ($< 1/n$)



- Inspired follow-up works [JW05,SW04,FW04,Sud05], but all regard rather artificial problems
  - *''It will take many major steps to prove rigorously that crossover is essential for typical applications.''* (Jansen and Wegener [JW05])
- [DHK08, DT09, DJK+13]: First *classic* optimization problem where crossover provably gives a speed-up (of around a factor of $\sqrt{n}$): All-pairs shortest path problem
  - rather specialized crossover operator

3rd Topic: Usefulness of Crossover (2/2)

- Possible insight (?): If it is so hard to find a single convincing problem where crossover provably helps, maybe crossover is not so important?

  → At least, crossover is not easy to get to work
- Interesting turning point: Unbiased black-box complexity model introduced by Lehre and Witt [GECCO 2010, Algorithmica 2012]
  - No mutation-based algorithm (formally, no unary unbiased black-box algorithm) can solve the OneMax problem faster than in $\Omega(n \ln n)$ iterations [LehreWitt GECCO'10 and Algorithmica'12, DoerrDoerrYang GECCO'16]
  - There is a binary unbiased black-box algorithm (still no cheating, but generating an offspring from two parents is now allowed) solving the OneMax problem in $2n$ iterations [DoerrJohannsenKötzingLehreWagnerWinzen FOGA'11]
    - This algorithm is NOT an EA, and possibly far from whatever we would consider as an EA.
    - BUT: it has inspired the development of a $\Theta(n)$ GA for OneMax

OneMax, or: The Hamming Distance Problem

- Unknown target vector $z \in \{0,1\}^n$
- Goal is to learn z
- "Fitness" of a search point x is "closeness of x to z", i.e., $n - H(x,z)$
- Mastermind game from the 70s

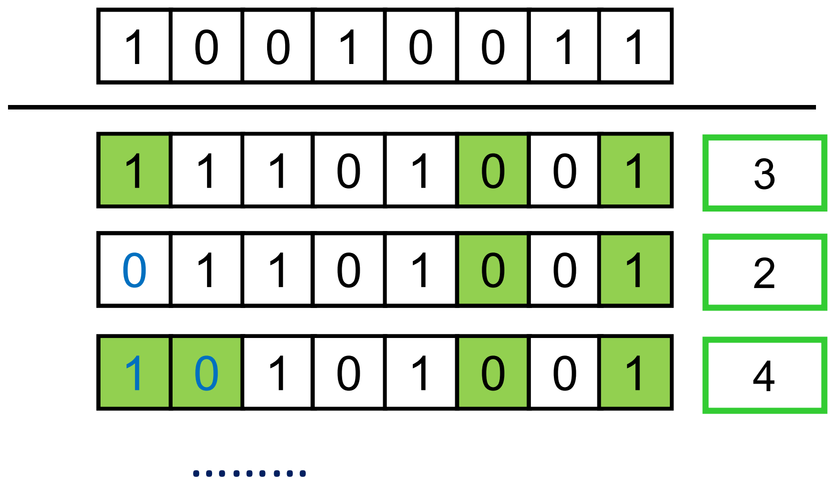
z = 1 1 1 1 1 0 0 0 0

x = 0 1 0 1 1 0 1 0 1

$f_z(x) = 9 - 4 = 5$

OneMax, or: The Hamming Distance Problem

- Unknown target vector $z \in \{0,1\}^n$ — z = 1 1 1 1 1 0 0 0 0
- Goal is to learn z
- "Fitness" of a search point x is "closeness of x to z", i.e., $n - H(x,z)$ — x = 0 1 0 1 1 0 1 0 1; $f_z(x) = 9 - 4 = 5$
- How long do you need to solve this problem?
- …
- Hopefully, less than n+1 queries:



OneMax, or: The Hamming Distance Problem

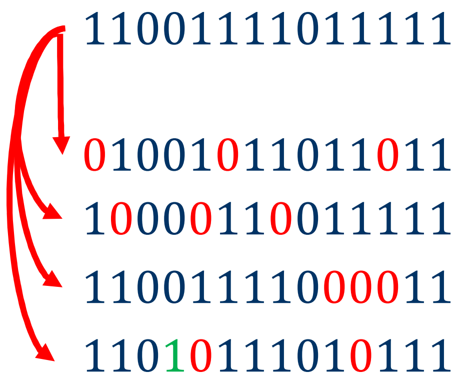
- Unknown target vector $z \in \{0,1\}^n$ — z = 1 1 1 1 1 0 0 0 0
- Goal is to learn z
- "Fitness" of a search point x is "closeness of x to z", i.e., $n - H(x,z)$ — x = 0 1 0 1 1 0 1 0 1; $f_z(x) = 9 - 4 = 5$
- How long do you need to solve this problem?
- …
- Hopefully, less than n+1 queries
- [Erdős & Rényi 1963]: one can do better: $O(n/\log n)$ strategy

Theory as Source for Inspiration

- Theory-driven curiosity: Explain the dichotomy between
  - theoretically best possible black-box algorithm $\mathcal{A}^*$ for OneMax needing only $O(n/\log n)$ fitness evaluations
  - all known EAs need $\Omega(n \ln n)$ fitness evaluations
- One explanation (from looking at the proofs of [Erdős & Rényi 1963] and [DoerrJohannsenKötzingLehreWagnerWinzen11]):
  - $o(n \log n)$ algorithms profit from *all* search points it generates,
  - whereas most EAs learn *only/mostly* from search points that are as good or better than the previous-best
- Can we invent an EA that also gains from inferior search points?
  - YES [DDE13,DD15a,DD15b,Doe16]: crossover-based algorithm needing $\Theta(n)$ function evaluations
    $\rightarrow$ best possible among all comparison-based algorithms

Using Crossover as A Repair Mechanism

- Assume that we are close to the optimum already
- When we create offspring at distance ℓ from x, the typical fitness of offspring is $f(x) - \ell$
- But there is some probability to sample offspring with fitness $> f(x) - \ell$
- In this offspring x’, one bit is correct that is not correct in x



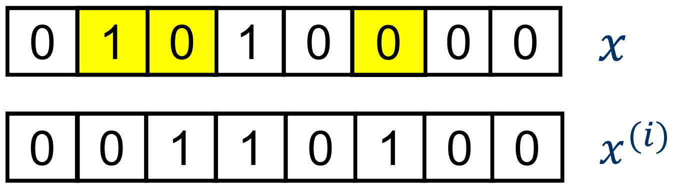
Using Crossover as A Repair Mechanism

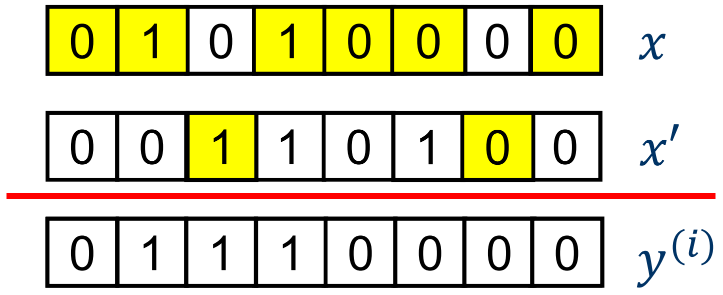
- Assume that we are close to the optimum already
- When we create offspring at distance ℓ from x, the typical fitness of offspring is $f(x) - \ell$
- But there is some probability to sample offspring with fitness $> f(x) - \ell$
- In this offspring x', one bit is correct that is not correct in x
- Hope: mechanism with
  - fair chance of extracting the "good" bit(s) of x'
  - not copying the "bad" bits of x'
- Uniform crossover: take entry from
  - x' with probability c
  - x with probability $1 - c$
- Uniform crossover existed before, novel is the usage <u>after</u> mutation

$$\begin{array}{cl} & 11001111011111 \\ + & 11010111010111 \\ \hline = & 11011111011111 \end{array}$$

The (1+(λ, λ)) GA

```
1.  Initialization: Sample $x \in \{0,1\}^n$ u.a.r.
2.  Optimization: for $t = 1,2,3,\ldots$ do
3.      Mutation phase:
4.          Sample $\ell$ from $B(n,p)$;
5.          for $i = 1,\ldots,\lambda$ do Sample $x^{(i)} \leftarrow \mathrm{mut}_\ell(x)$;
6.          Choose $x' \in \{x^{(1)},\ldots,x^{(\lambda)}\}$ with $f(x') = \max\{f(x^{(1)}),\ldots,f(x^{(\lambda)})\}$;
7.      Crossover phase:
8.          for $i = 1,\ldots,\lambda$ do Sample $y^{(i)} \leftarrow \mathrm{cross}_c(x,x')$;
9.          Choose $y \in \{y^{(1)},\ldots,y^{(\lambda)}\}$ with $f(y) = \max\{f(y^{(1)}),\ldots,f(y^{(\lambda)})\}$;
10.     Selection step: if $f(y) \geq f(x)$ then replace $x$ by $y$;
```





The (1+(λ, λ)) Genetic Algorithm

```
1.  Initialization: Sample $x \in \{0,1\}^n$ u.a.r.
2.  Optimization: for $t = 1,2,3, \dots$ do
3.    Mutation phase:
4.      Sample $\ell$ from $B(n, p)$;
5.      for $i = 1, \dots, \lambda$ do Sample $x^{(i)} \leftarrow \mathrm{mut}_\ell(x)$;
6.      Choose $x' \in \{x^{(1)}, \dots, x^{(\lambda)}\}$ with $f(x') = \max\{f(x^{(1)}), \dots, f(x^{(\lambda)})\}$;
7.    Crossover phase:
8.      for $i = 1, \dots, \lambda$ do Sample $y^{(i)} \leftarrow \mathrm{cross}_c(x, x')$;
9.      Choose $y \in \{y^{(1)}, \dots, y^{(\lambda)}\}$ with $f(y) = \max\{f(y^{(1)}), \dots, f(y^{(\lambda)})\}$;
10.   Selection step: if $f(y) \geq f(x)$ then replace $x$ by $y$;
```

How to choose λ, p, and c?

The (1+(λ, λ)) Genetic Algorithm

```
1.  Initialization: Sample $x \in \{0,1\}^n$ u.a.r.
2.  Optimization: for $t = 1,2,3,\ldots$ do
3.    Mutation phase:
4.      Sample $\ell$ from $B(n, \lambda/n)$;
5.      for $i = 1,\ldots,\lambda$ do Sample $x^{(i)} \leftarrow \mathrm{mut}_\ell(x)$;
6.      Choose $x' \in \{x^{(1)},\ldots,x^{(\lambda)}\}$ with $f(x') = \max\{f(x^{(1)}),\ldots,f(x^{(\lambda)})\}$;
7.    Crossover phase:
8.      for $i = 1,\ldots,\lambda$ do Sample $y^{(i)} \leftarrow \mathrm{cross}_{1/\lambda}(x, x')$;
9.      Choose $y \in \{y^{(1)},\ldots,y^{(\lambda)}\}$ with $f(y) = \max\{f(y^{(1)}),\ldots,f(y^{(\lambda)})\}$;
10.   Selection step: if $f(y) \geq f(x)$ then replace $x$ by $y$;
```

How to choose λ, p, and c?

➔ $p = \lambda/n$, $c = 1/\lambda$ *look reasonable*

(and we can now even prove that they are optimal)

➔ only one parameter left

Performance of the (1+(λ, λ)) GA

- We first showed that the expected runtime of the (1+(λ, λ)) GA on OneMax is $O(n \log n/\lambda + \lambda n)$ [DoerrDoerrEbel15]
- With this bound, we get the best results for $\lambda = \Theta\left(\sqrt{\log n}\right)$:
  runtime is $O\left(n\sqrt{\log n}\right)$
- → <u>**Key Message:**</u> first time a GA with an asymptotic gain over standard EAs is proven for a simple fitness landscape such as OneMax
  - Improvement over "classic" $\Theta(n \log n)$ bound for quite a range of different values for λ
- We later improved this bound to $\Theta(n \sqrt{\log n \log\log\log n} / \sqrt{\log\log n})$ [DoerrDoerr15] and showed that this is optimal for all static parameter choices [Doerr16]
  - Still not linear, but better than previous EAs and better than any (!) mutation-based algorithm

Discussion of the Runtime Bound

- Observation (follows from the analysis of the algorithm):
  a dynamic, fitness-dependent rate of λ would yield linear time
  - More precisely, the expected running time of the (1+(λ, λ)) GA with $\lambda = \sqrt{\frac{n}{n-f(x)}}$ is linear in n
  - ☺ Better than any static parameter setting
  - ☺ Linear bound seems "natural" for this problem and is best possible for all comparison-based algos
  - ☹ Requires some good guessing or a good understanding of the optimization process (this is not doable for more complex problems)

Key research question:

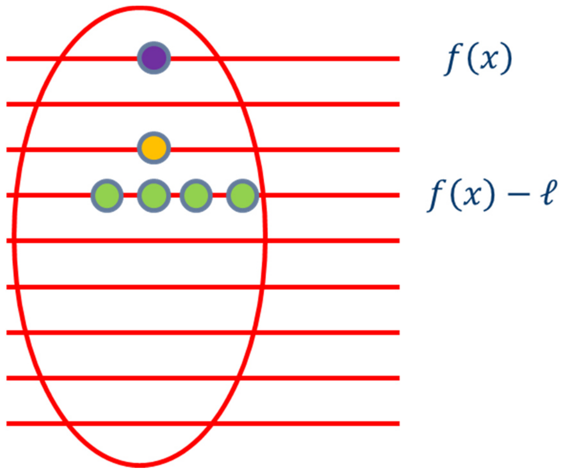
Is there a simpler way to get this linear performance?

YES:

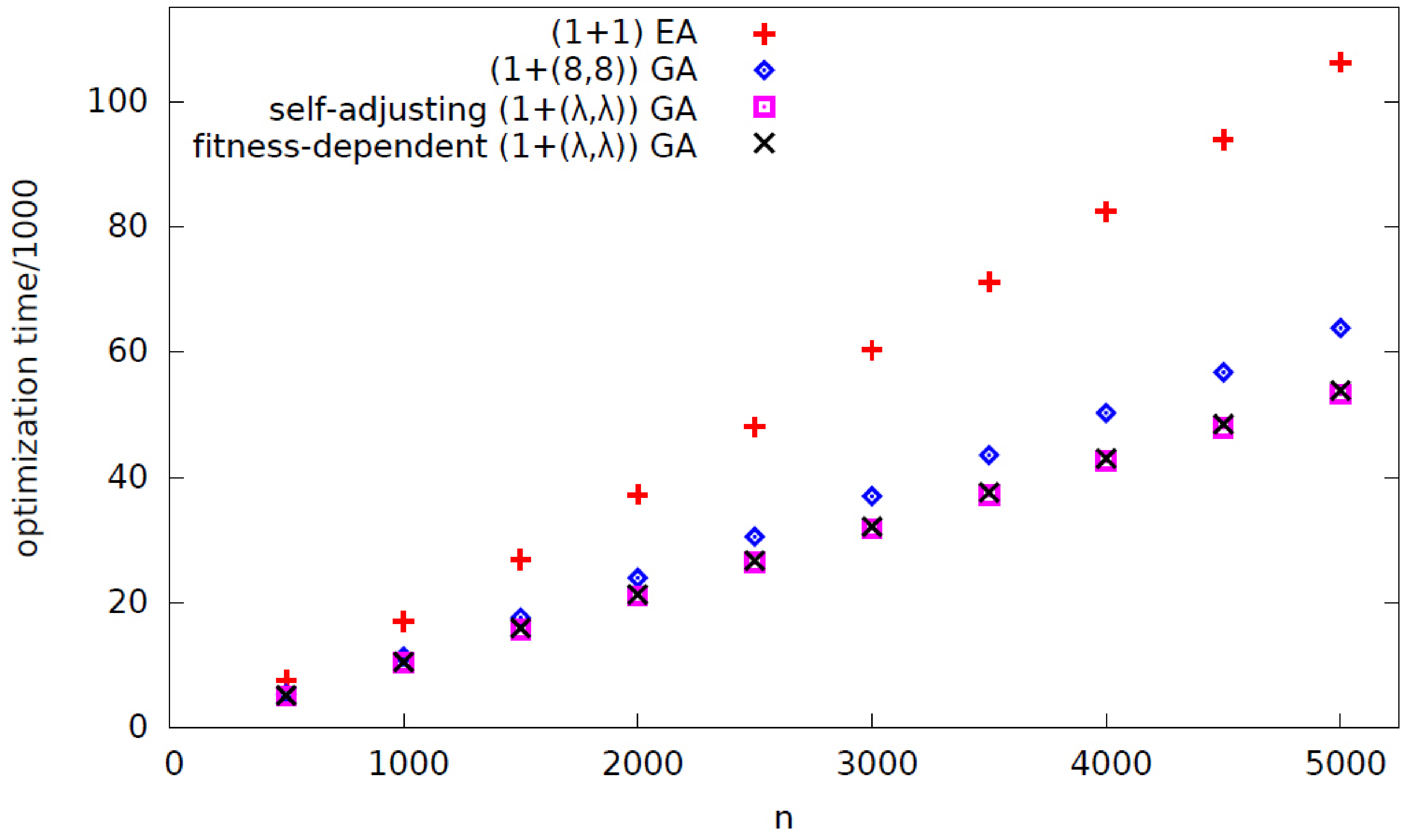
Self-adjusting choice of λ

4th Topic: Self-Adjusting Parameter Choices

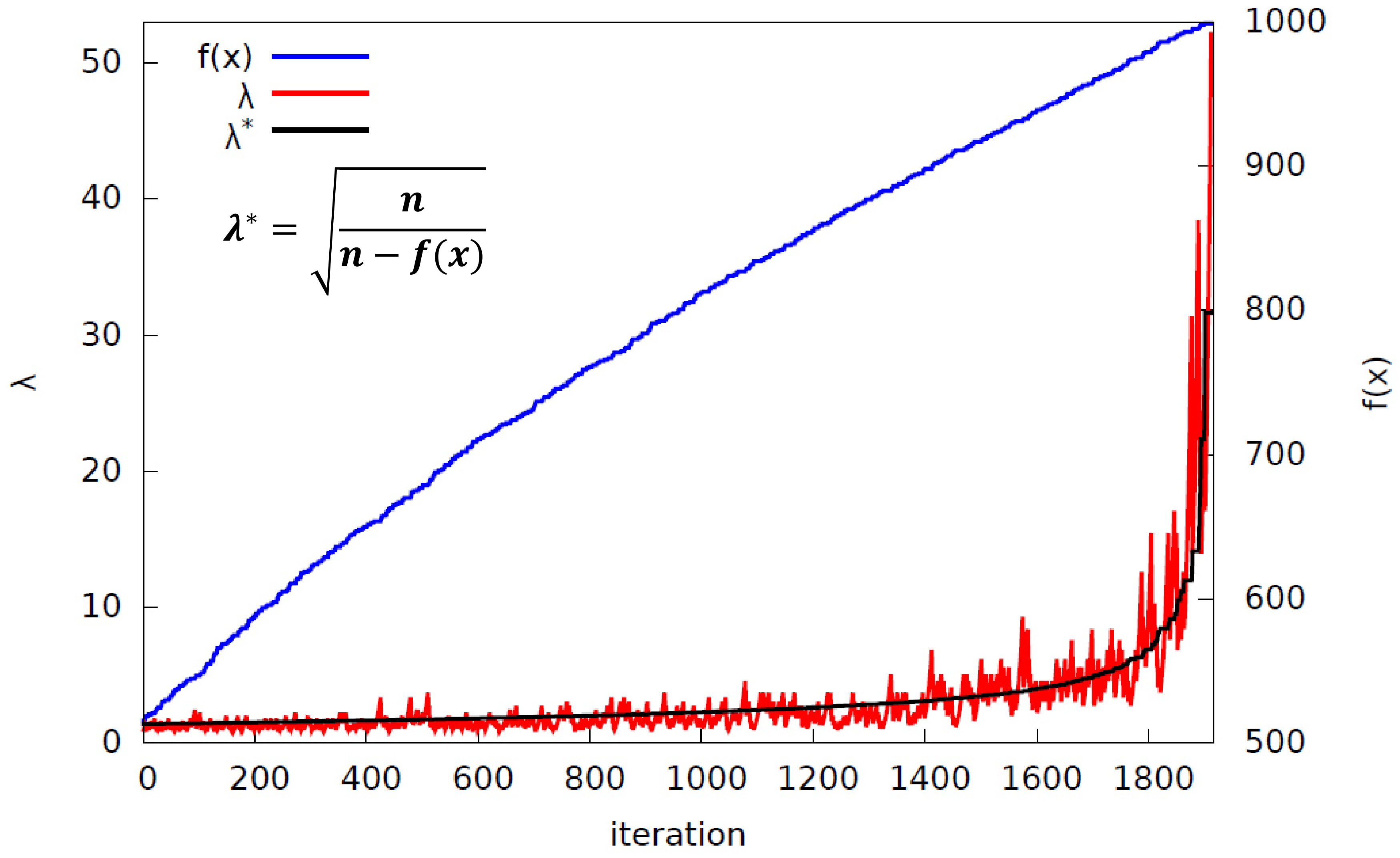
- Optimal dynamic parameter: $\lambda = \sqrt{\frac{n}{n-f(x)}}$
- Can the algorithm find this optimal setting for λ itself?
- Idea: implement a so-called 1/5th success rule:
  - If at the end of an iteration
    - we have an improvement ($f(y) > f(x)$) then $\lambda \leftarrow \lambda/F$;
    - No improvement ($f(y) \leq f(x)$) then $\lambda \leftarrow \lambda F^{1/4}$;
  - Intuition:
    - if we had an improvement, we may have spent too much for it, so let's try to get improvements using smaller λ
    - If we did not see an improvement we probably need to increase the chances for sampling good offspring and for a successful crossover



Experimental Results for Self-Adjusting Version



Example Run Self-Adjusting $(1 + (\lambda, \lambda))$ GA

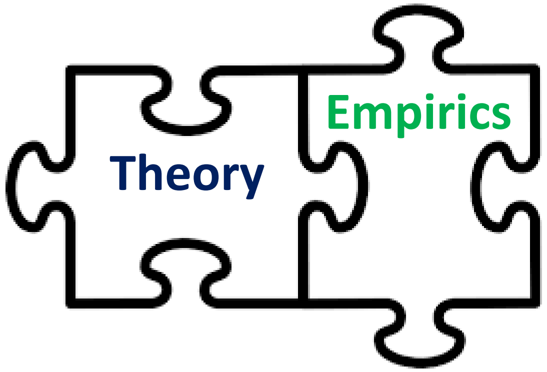


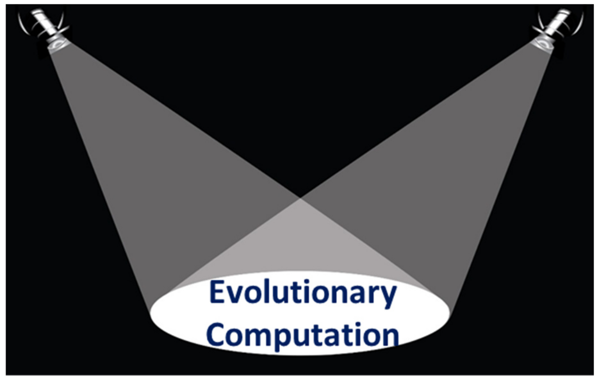
Runtime Result for the Self-Adjusting GA

- Theorem: The self-adjusting (1+(λ, λ)) GA has $\Theta(n)$ expected runtime on OneMax (for sufficiently small update strength F>1). [Doerr, Doerr GECCO'15]
- We can also prove that this is optimal among all possible parameter settings. And it is optimal for comparison-based algorithms.
- General insights:
  - The constant F matters, but not too much
  - The adjustment rule (1/5 vs 1/4 vs 1/6 vs…) must fit to the limiting success probability

Towards a More Implementation-Aware Theory

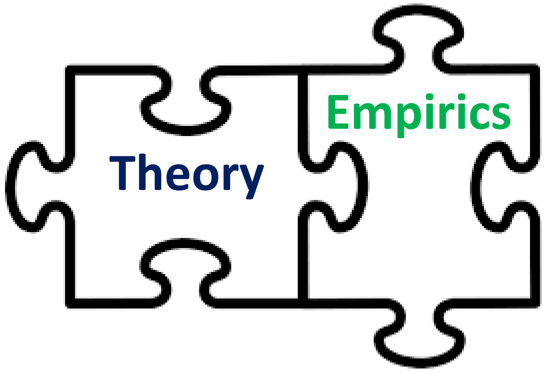
- We have said that theory and empirics complement each other:

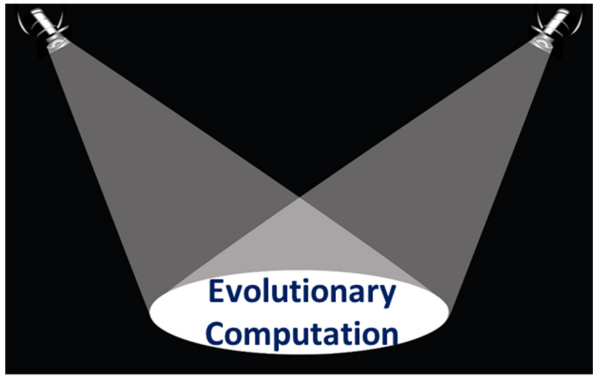




Towards a More Implementation-Aware Theory

- We have said that theory and empirics complement each other:





- A practice-aware theory can shed new light on old findings:
  - [Sudholt 12]: The Greedy (2+1) GA is better than any EA using standard bit mutation only.
  - [Carvalho Pinto, Doerr 17+]: The Greedy (2+1) GA is better than any unary unbiased algorithm, *if we do not count function evaluations in which the offspring equals one of its parents.*

Towards a More Implementation-Aware Theory

- A small difference in the pseudo code…

Algorithm 1: The well-known $(1+1)$ EA with mutation probability $p \in (0,1)$ for the maximization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$

```
Initialization: Sample $x \in \{0,1\}^n$ uniformly at random and compute $f(x)$;
Optimization: for $t = 1, 2, 3, \dots$ do
    Sample $\ell \sim \mathrm{Bin}(n, p)$;
    $y \leftarrow \mathrm{mut}_\ell(x)$;
    evaluate $f(y)$;
    if $f(y) \geq f(x)$ then $x \leftarrow y$;
```

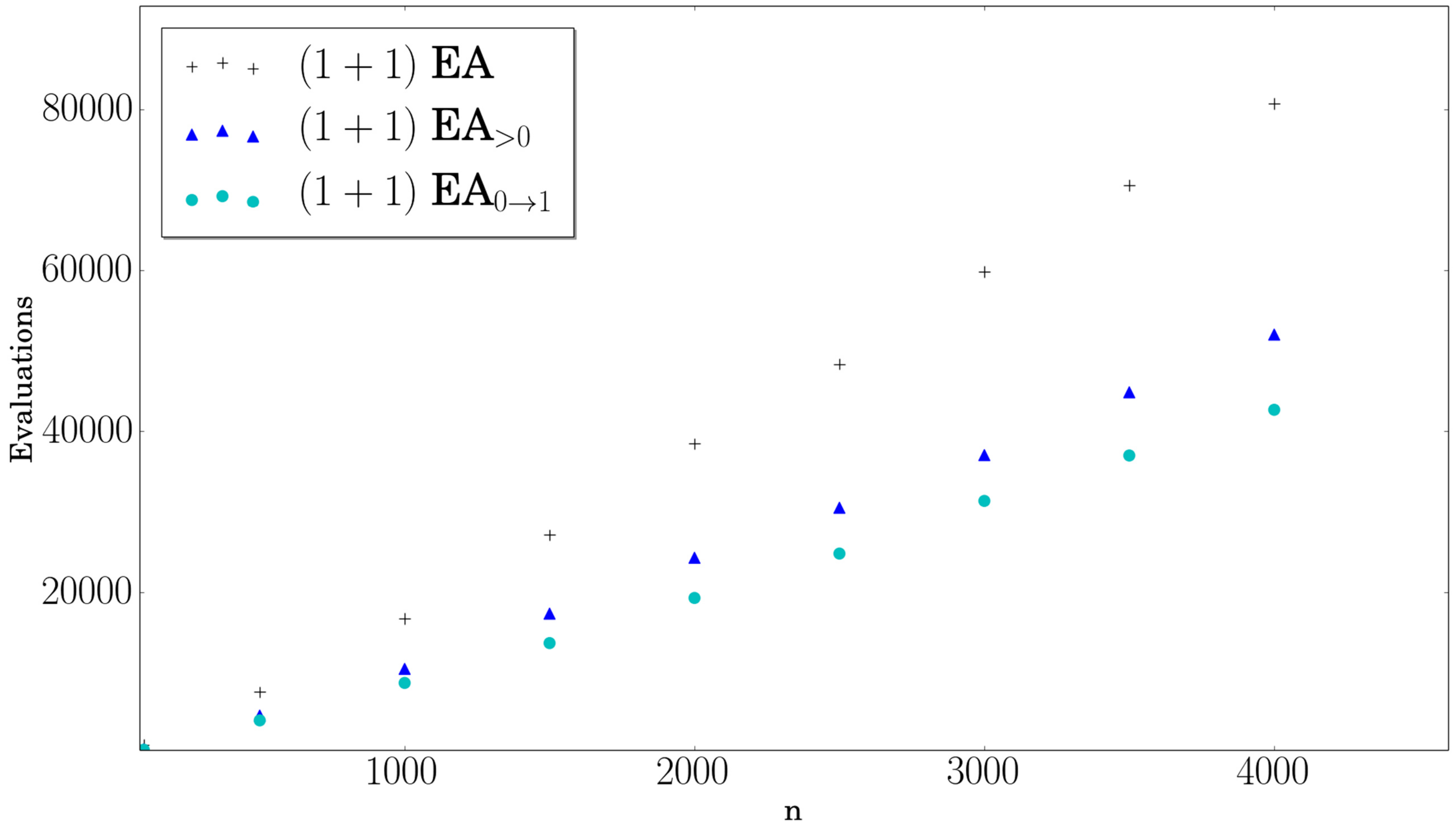
Algorithm 3: The $(1+1)$ EA$_{>0}$ with mutation probability $p \in (0,1)$ for the maximization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$

```
Initialization: Sample $x \in \{0,1\}^n$ uniformly at random and compute $f(x)$;
Optimization: for $t = 1, 2, 3, \dots$ do
    Sample $\ell \sim \mathrm{Bin}_{>0}(n, p)$;
    $y \leftarrow \mathrm{mut}_\ell(x)$;
    evaluate $f(y)$;
    if $f(y) \geq f(x)$ then $x \leftarrow y$;
```

Towards a More Implementation-Aware Theory

- A small difference in the pseudo code…

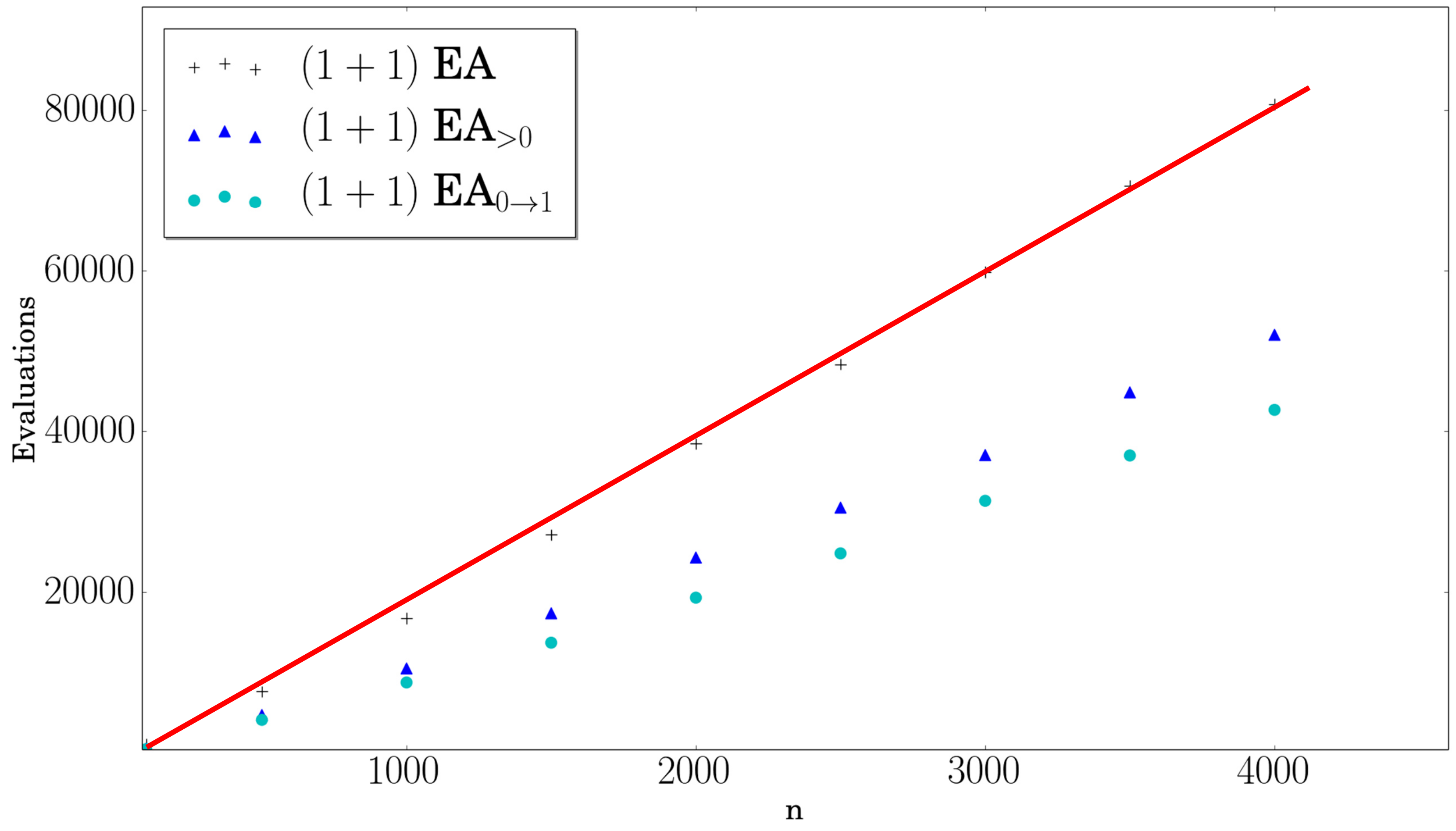
  … can make a huge difference in performance



Towards a More Implementation-Aware Theory

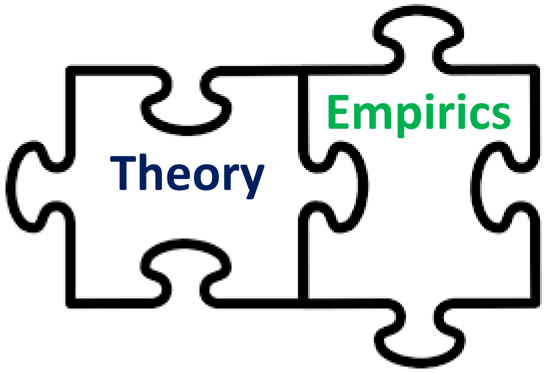
- A small difference in the pseudo code…

  … can make a huge difference in performance



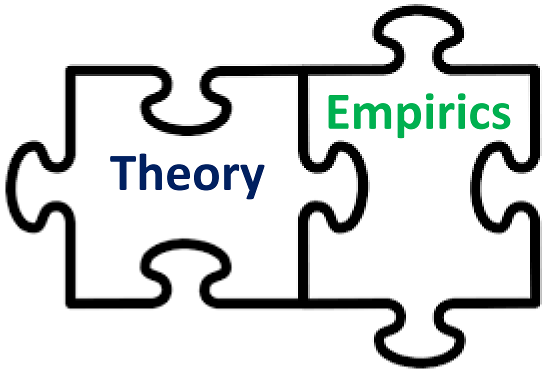
Take-Home Messages

1. Whether you use a rather empirical or a theoretical approach in your research, you are advised to follow what the “other side” is doing



Take-Home Messages

1. Whether you use a rather empirical or a theoretical approach in your research, you are advised to follow what the "other side" is doing



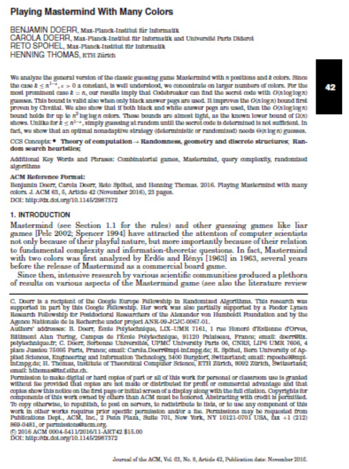
2. Title of this talk: **<u>Theory of Discrete *Black-Box Optimization*</u>**
→ because EC research suffers from not being overly aware of what is going on in neighboring fields (artificial intelligence, optimization, operations research, machine learning, theoretical computer science, etc.)
→ Try to be very alert and try to connect your research to these fields!

3 Papers for Summaries

1. Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. Playing Mastermind with Many Colors. *Journal of the ACM*, Volume 63, Article 42, 2016.

   → How theory of EC can be quite fun

   → (and how it can yield surprising insights in "classical" theoretical computer science research)



Playing Mastermind With Many Colors

BENJAMIN DOERR, Max-Planck-Institut für Informatik
CAROLA DOERR, Max-Planck-Institut für Informatik and Université Paris Diderot
RETO SPÖHEL, Max-Planck-Institut für Informatik
HENNING THOMAS, ETH Zürich

42

We analyze the general version of the classic guessing game Mastermind with n positions and k colors. Since the case $k \leq n^{1-\varepsilon}$, $\varepsilon > 0$ a constant, is well understood, we concentrate on larger numbers of colors. For the most prominent case $k = n$, our results imply that Codebreaker can find the secret code with $O(n \log \log n)$ guesses. This bound is valid also when only black answer pegs are used. It improves the $O(n \log n)$ bound first proven by Chvátal. We also show that if both black and white answer pegs are used, then the $O(n \log \log n)$ bound holds for up to $n^2 \log \log n$ colors. These bounds are almost tight, as the known lower bound of $\Omega(n)$ shows. Unlike for $k \leq n^{1-\varepsilon}$, simply guessing at random until the secret code is determined is not sufficient. In fact, we show that an optimal nonadaptive strategy (deterministic or randomized) needs $\Theta(n \log n)$ guesses.

CCS Concepts: • **Theory of computation → Randomness, geometry and discrete structures; Random search heuristics;**

Additional Key Words and Phrases: Combinatorial games, Mastermind, query complexity, randomized algorithms

ACM Reference Format:
Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. 2016. Playing Mastermind with many colors. J. ACM 63, 5, Article 42 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2987372

1. INTRODUCTION

Mastermind (see Section 1.1 for the rules) and other guessing games like liar games [Pelc 2002; Spencer 1994] have attracted the attention of computer scientists not only because of their playful nature, but more importantly because of their relation to fundamental complexity and information-theoretic questions. In fact, Mastermind with two colors was first analyzed by Erdős and Rényi [1963] in 1963, several years before the release of Mastermind as a commercial board game.

Since then, intensive research by various scientific communities produced a plethora of results on various aspects of the Mastermind game (see also the literature review

C. Doerr is a recipient of the Google Europe Fellowship in Randomized Algorithms. This research was supported in part by this Google Fellowship. Her work was also partially supported by a Feodor Lynen Research Fellowship for Postdoctoral Researchers of the Alexander von Humboldt Foundation and by the Agence Nationale de la Recherche under project ANR-09-JCJC-0067-01.

Authors' addresses: B. Doerr, École Polytechnique, LIX–UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bâtiment Alan Turing, Campus de l'École Polytechnique, 91120 Palaiseau, France; email: doerr@lix.polytechnique.fr; C. Doerr, Sorbonne Universités, UPMC University Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France; email: Carola.Doerr@mpi-inf.mpg.de; R. Spöhel, Bern University of Applied Sciences, Engineering and Information Technology, 3400 Burgdorf, Switzerland; email: rspoehel@mpi-inf.mpg.de; H. Thomas, Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland; email: hthomas@inf.ethz.ch.

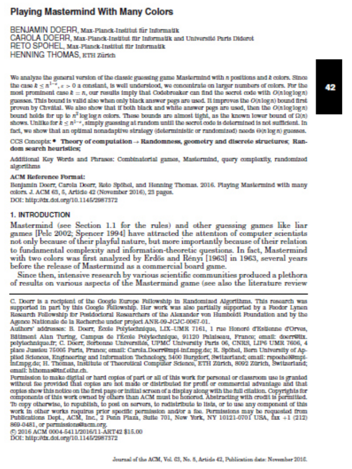
DOI: http://dx.doi.org/10.1145/2987372

3 Papers for Summaries

1. Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. Playing Mastermind with Many Colors. *Journal of the ACM*, Volume 63, Article 42, 2016.

   → *How theory of EC can be quite fun*

   → *(and how it can yield surprising insights in "classical" theoretical computer science research)*



Playing Mastermind With Many Colors

BENJAMIN DOERR, Max-Planck-Institut für Informatik
CAROLA DOERR, Max-Planck-Institut für Informatik and Universitéé Paris Diderot
RETO SPÖHEL, Max-Planck-Institut für Informatik
HENNING THOMAS, ETH Zürich

42

We analyze the general version of the classic guessing game Mastermind with n positions and k colors. Since the case $k \le n^{1-\varepsilon}$, $\varepsilon > 0$ a constant, is well understood, we concentrate on larger numbers of colors. For the most prominent case $k = n$, our results imply that Codebreaker can find the secret code with $O(n \log\log n)$ guesses. This bound is valid also when only black answer pegs are used. It improves the $O(n \log n)$ bound first proven by Chvátal. We also show that if both black and white answer pegs are used, then the $O(n \log\log n)$ bound holds for up to $n^2 \log\log n$ colors. These bounds are almost tight, as the known lower bound of $\Omega(n)$ shows. Unlike for $k \le n^{1-\varepsilon}$, simply guessing at random until the secret code is determined is not sufficient. In fact, we show that an optimal nonadaptive strategy (deterministic or randomized) needs $\Theta(n \log n)$ guesses.

CCS Concepts: • **Theory of computation → Randomness, geometry and discrete structures; Random search heuristics;**

Additional Key Words and Phrases: Combinatorial games, Mastermind, query complexity, randomized algorithms

ACM Reference Format:
Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. 2016. Playing Mastermind with many colors. J. ACM 63, 5, Article 42 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2987372

1. INTRODUCTION

Mastermind (see Section 1.1 for the rules) and other guessing games like liar games [Pelc 2002; Spencer 1994] have attracted the attention of computer scientists not only because of their playful nature, but more importantly because of their relation to fundamental complexity and information-theoretic questions. In fact, Mastermind with two colors was first analyzed by Erdős and Rényi [1963] in 1963, several years before the release of Mastermind as a commercial board game.

Since then, intensive research by various scientific communities produced a plethora of results on various aspects of the Mastermind game (see also the literature review

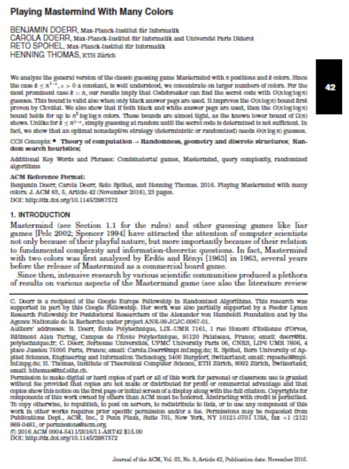
C. Doerr is a recipient of the Google Europe Fellowship in Randomized Algorithms. This research was supported in part by this Google Fellowship. Her work was also partially supported by a Feodor Lynen Research Fellowship for Postdoctoral Researchers of the Alexander von Humboldt Foundation and by the Agence Nationale de la Recherche under project ANR-09-JCJC-0067-01.
Authors' addresses: B. Doerr, École Polytechnique, LIX–UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bâtiment Alan Turing, Campus de l'École Polytechnique, 91120 Palaiseau, France; email: doerr@lix.polytechnique.fr; C. Doerr, Sorbonne Universités, UPMC University Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France; email: Carola.Doerr@mpi-inf.mpg.de; R. Spöhel, Bern University of Applied Sciences, Engineering and Information Technology, 3400 Burgdorf, Switzerland; email: rspoehel@mpi-inf.mpg.de; H. Thomas, Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland; email: hthomas@inf.ethz.ch.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 0004-5411/2016/11-ART42 $15.00
DOI: http://dx.doi.org/10.1145/2987372

Journal of the ACM, Vol. 63, No. 5, Article 42, Publication date: November 2016.

2. Benjamin Doerr, Carola Doerr, and Franziska Ebel. From Black-Box Complexity to Designing New Genetic Algorithms. *Theoretical Computer Science*, Volume 567, pages 87-104, 2015.

   → *How Theory can inspire new algorithms*

3 Papers for Summaries



Playing Mastermind With Many Colors

BENJAMIN DOERR, Max-Planck-Institut für Informatik
CAROLA DOERR, Max-Planck-Institut für Informatik and Université Paris Diderot
RETO SPÖHEL, Max-Planck-Institut für Informatik
HENNING THOMAS, ETH Zürich

42

We analyze the general version of the classic guessing game Mastermind with n positions and k colors. Since the case $k \leq n^{1-\varepsilon}$, $\varepsilon > 0$ a constant, is well understood, we concentrate on larger numbers of colors. For the most prominent case $k = n$, our results imply that Codebreaker can find the secret code with $O(n \log \log n)$ guesses. This bound is valid also when only black answer pegs are used. It improves the $O(n \log n)$ bound first proven by Chvátal. We also show that if both black and white answer pegs are used, then the $O(n \log \log n)$ bound holds for up to $n^2 \log \log n$ colors. These bounds are almost tight, as the known lower bound of $\Omega(n)$ shows. Unlike for $k \leq n^{1-\varepsilon}$, simply guessing at random until the secret code is determined is not sufficient. In fact, we show that an optimal nonadaptive strategy (deterministic or randomized) needs $\Theta(n \log n)$ guesses.

CCS Concepts: • **Theory of computation → Randomness, geometry and discrete structures; Random search heuristics;**

Additional Key Words and Phrases: Combinatorial games, Mastermind, query complexity, randomized algorithms

ACM Reference Format:
Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. 2016. Playing Mastermind with many colors. J. ACM 63, 5, Article 42 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2987372

1. INTRODUCTION

Mastermind (see Section 1.1 for the rules) and other guessing games like liar games [Pelc 2002; Spencer 1994] have attracted the attention of computer scientists not only because of their playful nature, but more importantly because of their relation to fundamental complexity and information-theoretic questions. In fact, Mastermind with two colors was first analyzed by Erdős and Rényi [1963] in 1963, several years before the release of Mastermind as a commercial board game.

Since then, intensive research by various scientific communities produced a plethora of results on various aspects of the Mastermind game (see also the literature review

C. Doerr is a recipient of the Google Europe Fellowship in Randomized Algorithms. This research was supported in part by this Google Fellowship. Her work was also partially supported by a Feodor Lynen Research Fellowship for Postdoctoral Researchers of the Alexander von Humboldt Foundation and by the Agence Nationale de la Recherche under project ANR-09-JCJC-0067-01.
Authors' addresses: B. Doerr, École Polytechnique, LIX–UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bâtiment Alan Turing, Campus de l'École Polytechnique, 91120 Palaiseau, France; email: doerr@lix.polytechnique.fr; C. Doerr, Sorbonne Universités, UPMC University Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France; email: Carola.Doerr@mpi-inf.mpg.de; R. Spöhel, Bern University of Applied Sciences, Engineering and Information Technology, 3400 Burgdorf, Switzerland; email: rspoehel@mpi-inf.mpg.de; H. Thomas, Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland; email: hthomas@inf.ethz.ch.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 0004-5411/2016/11-ART42 $15.00
DOI: http://dx.doi.org/10.1145/2987372

Journal of the ACM, Vol. 63, No. 5, Article 42, Publication date: November 2016.

1. Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. Playing Mastermind with Many Colors. *Journal of the ACM*, Volume 63, Article 42, 2016.

   → *How theory of EC can be quite fun*

   → *(and how it can yield surprising insights in "classical" theoretical computer science research)*

2. Benjamin Doerr, Carola Doerr, and Franziska Ebel. From Black-Box Complexity to Designing New Genetic Algorithms. *Theoretical Computer Science*, Volume 567, pages 87-104, 2015.

   → *How Theory can inspire new algorithms*

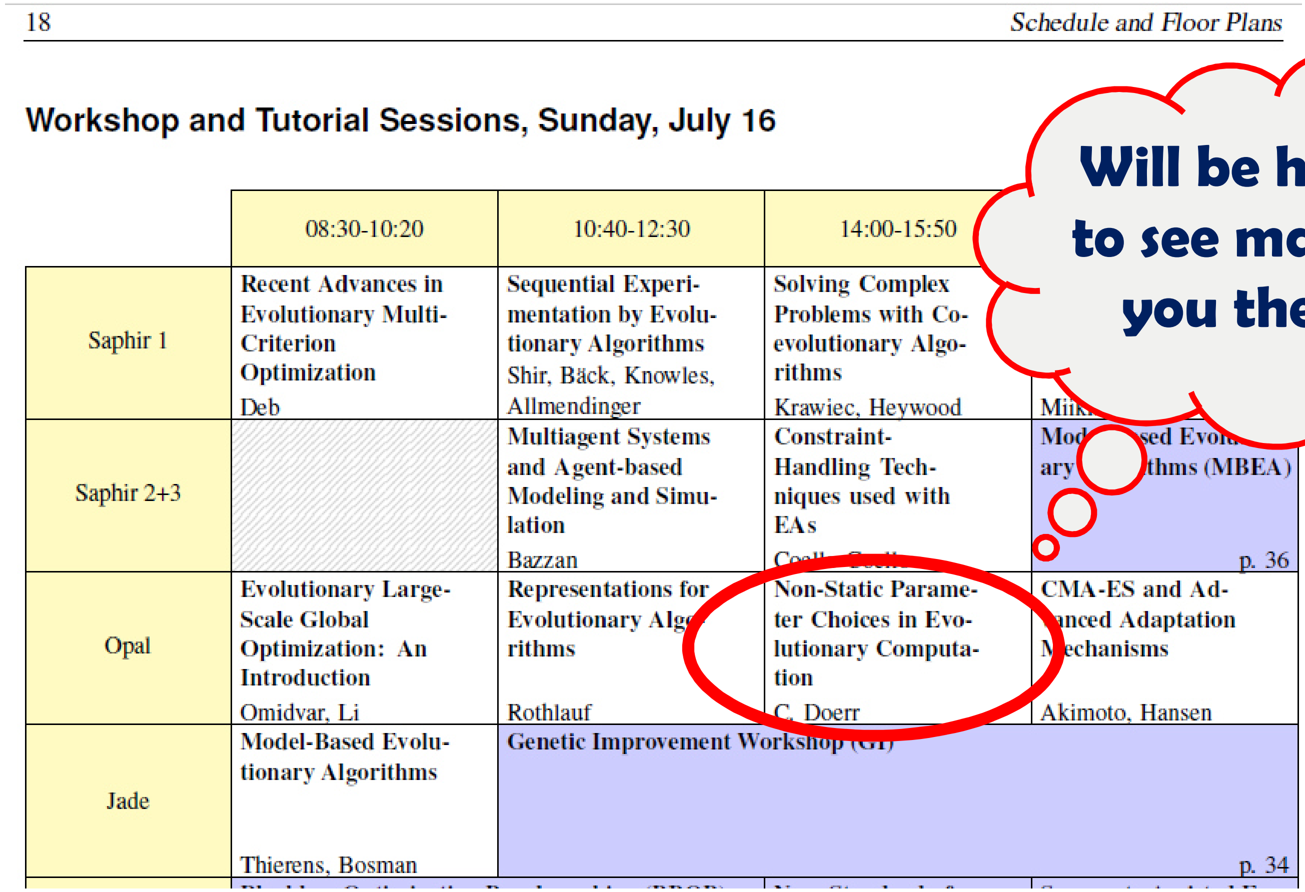
3. Benjamin Doerr, and Carola Doerr. Theory for Non-Theoreticians. Slides of a tutorial held at GECCO 2016. *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2016)*, pages 463-482, ACM, 2016.

   → *What theory of EAs is about*

Your Challenge

Related to Self-Adjusting Parameter Choices

→ Tutorial on *Non-Static Parameter Choices in EC*, Sunday afternoon, 14:00-15:50, room Opal



Workshop and Tutorial Sessions, Sunday, July 16

	08:30-10:20	10:40-12:30	14:00-15:50	
Saphir 1	**Recent Advances in Evolutionary Multi-Criterion Optimization** Deb	**Sequential Experimentation by Evolutionary Algorithms** Shir, Bäck, Knowles, Allmendinger	**Solving Complex Problems with Coevolutionary Algorithms** Krawiec, Heywood	Mii
Saphir 2+3	▨	**Multiagent Systems and Agent-based Modeling and Simulation** Bazzan	**Constraint-Handling Techniques used with EAs** Coello Coello	**Mod sed Evol ary thms (MBEA)** p. 36
Opal	**Evolutionary Large-Scale Global Optimization: An Introduction** Omidvar, Li	**Representations for Evolutionary Alg rithms** Rothlauf	**Non-Static Parameter Choices in Evolutionary Computation** C. Doerr	**CMA-ES and Ad anced Adaptation echanisms** Akimoto, Hansen
Jade	**Model-Based Evolutionary Algorithms** Thierens, Bosman	**Genetic Improvement Workshop (GI)** p. 34		

Will be happy to see many of you there !

Assignments: Self-Adjusting Parameter Choices

- **Part A) Literature Survey**
- **Part B) Empirical Study**

Assignments: Self-Adjusting Parameter Choices

Part A) Literature Survey: Summarize and compare the principal approaches to control the parameters of an EA proposed in the following four papers (I can send them to you by e-mail if you do not have access to them).

- Álvaro Fialho, Luís Da Costa, Marc Schoenauer, Michèle Sebag: *Analyzing bandit-based adaptive operator selection mechanisms.* Ann. Math. Artif. Intell. 60(1-2): 25-64 (2010)

- Dirk Thierens: *An adaptive pursuit strategy for allocating operator probabilities.* GECCO 2005: 1539-1546

- Benjamin Doerr, Carola Doerr, Jing Yang: *Optimal Parameter Choices via Precise Black-Box Analysis.* GECCO 2016: 1123-1130

- Benjamin Doerr, Carola Doerr: *Optimal Parameter Choices Through Self-Adjustment: Applying the 1/5-th Rule in Discrete Settings.* GECCO 2015: 1335-1342

(Feel free to add other adaptation mechanisms to this comparison, if you find another one that catches your interest.)

Assignments: Self-Adjusting Parameter Choices

B) Empirical Study: Chose a benchmark problem and an algorithmic framework that you find interesting and test the different parameter control mechanisms.

- ***Examples for benchmark problems***
  (you can also use continuous benchmark problems, if you prefer):
  - "toy problems" like OneMax, LeadingOnes, linear functions, etc.
  - combinatorial problems like MaxSAT, Minimum Spanning Trees, etc.
  - existing benchmark suits
- ***Examples for algorithmic frameworks:***
  - simple stochastic search like (1+1) EA with adaptive mutation strengths, local search with varying neighborhood size, etc
  - population-based methods
  - swarm algorithms
  - etc.

Report your findings in a suitable way (i.e., using plots and text to describe performance for different problem dimensions and/or fitness evolution over time). *How do they compare to static parameter settings?* Please also describe the difficulties in implementing the different strategies.

Assignments: Self-Adjusting Parameter Choices

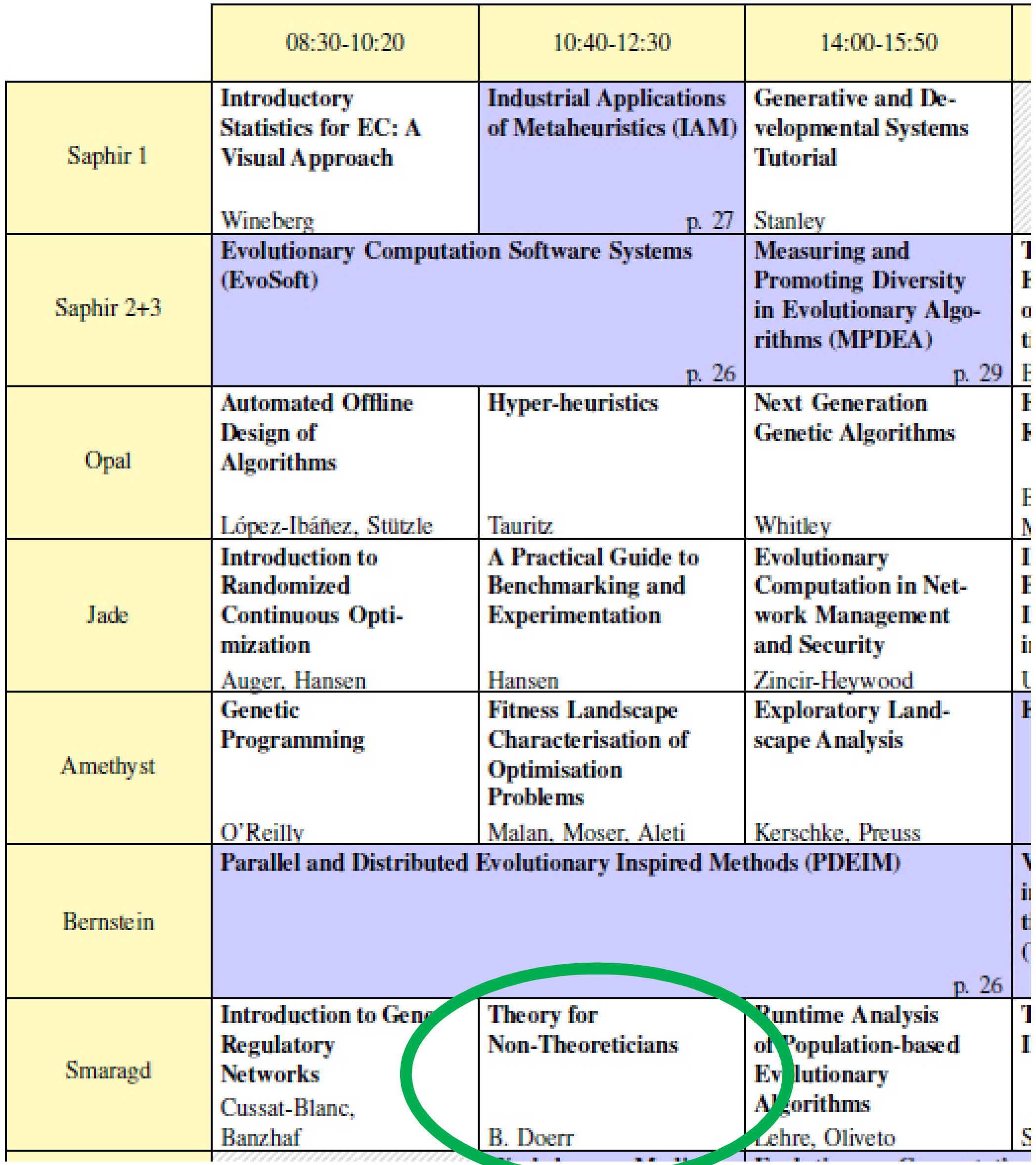
Deliverables: short report summarizing Parts A and B. A short presentation of your work.

Learning Goals:

- What are advantages of parameter control over parameter tuning?
- Which mechanisms have been experimented with? (obviously, you will see a tiny part only, but this hopefully inspires you to research more!)
- How easy (or not) is it to get parameter control going?
- How sensitive are the different methods with respect to the hyper-parameters?
- How to conduct well-designed experiments
- How to report your findings

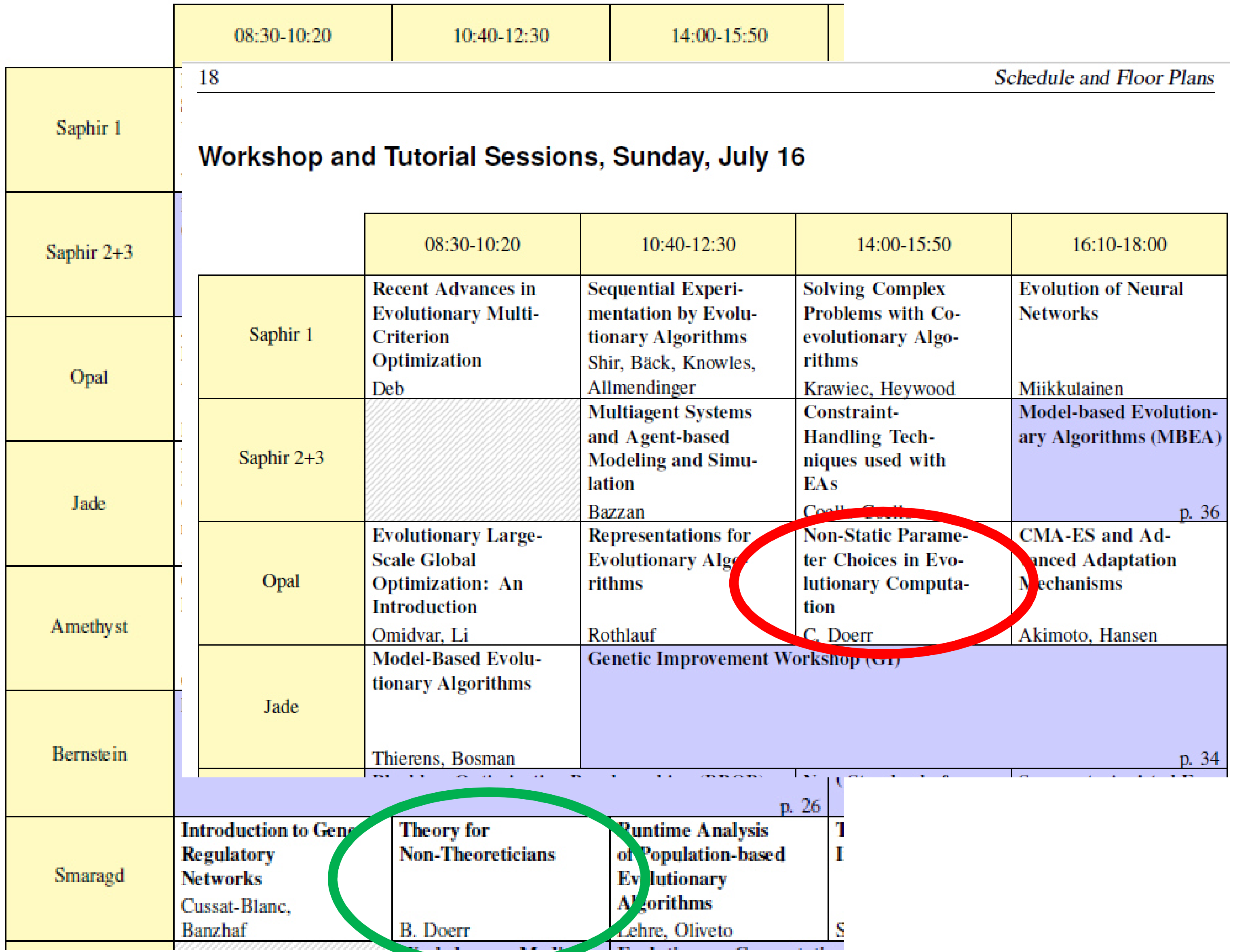
» Hopefully inspires you to explore more!

If you want to know more….



	08:30-10:20	10:40-12:30	14:00-15:50	
Saphir 1	**Introductory Statistics for EC: A Visual Approach** Wineberg	**Industrial Applications of Metaheuristics (IAM)** p. 27	**Generative and Developmental Systems Tutorial** Stanley	
Saphir 2+3	**Evolutionary Computation Software Systems (EvoSoft)** p. 26		**Measuring and Promoting Diversity in Evolutionary Algorithms (MPDEA)** p. 29	T F o t E
Opal	**Automated Offline Design of Algorithms** López-Ibáñez, Stützle	**Hyper-heuristics** Tauritz	**Next Generation Genetic Algorithms** Whitley	F F E N
Jade	**Introduction to Randomized Continuous Optimization** Auger, Hansen	**A Practical Guide to Benchmarking and Experimentation** Hansen	**Evolutionary Computation in Network Management and Security** Zincir-Heywood	I F I i U
Amethyst	**Genetic Programming** O'Reilly	**Fitness Landscape Characterisation of Optimisation Problems** Malan, Moser, Aleti	**Exploratory Landscape Analysis** Kerschke, Preuss	F
Bernstein	**Parallel and Distributed Evolutionary Inspired Methods (PDEIM)** p. 26			V i t (
Smaragd	**Introduction to Gene Regulatory Networks** Cussat-Blanc, Banzhaf	**Theory for Non-Theoreticians** B. Doerr	**Runtime Analysis of Population-based Evolutionary Algorithms** Lehre, Oliveto	T I S

If you want to know more….



	08:30-10:20	10:40-12:30	14:00-15:50
Saphir 1			
Saphir 2+3			
Opal			
Jade			
Amethyst			
Bernstein		p. 26	
Smaragd	**Introduction to Gen Regulatory Networks** Cussat-Blanc, Banzhaf	**Theory for Non-Theoreticians** B. Doerr	**Runtime Analysis of Population-based Evolutionary Algorithms** Lehre, Oliveto

18 *Schedule and Floor Plans*

Workshop and Tutorial Sessions, Sunday, July 16

	08:30-10:20	10:40-12:30	14:00-15:50	16:10-18:00
Saphir 1	**Recent Advances in Evolutionary Multi-Criterion Optimization** Deb	**Sequential Experimentation by Evolutionary Algorithms** Shir, Bäck, Knowles, Allmendinger	**Solving Complex Problems with Coevolutionary Algorithms** Krawiec, Heywood	**Evolution of Neural Networks** Miikkulainen
Saphir 2+3	▨	**Multiagent Systems and Agent-based Modeling and Simulation** Bazzan	**Constraint-Handling Techniques used with EAs** Coello Coello	**Model-based Evolutionary Algorithms (MBEA)** p. 36
Opal	**Evolutionary Large-Scale Global Optimization: An Introduction** Omidvar, Li	**Representations for Evolutionary Algorithms** Rothlauf	**Non-Static Parameter Choices in Evolutionary Computation** C. Doerr	**CMA-ES and Advanced Adaptation Mechanisms** Akimoto, Hansen
Jade	**Model-Based Evolutionary Algorithms** Thierens, Bosman	**Genetic Improvement Workshop (GI)** p. 34		

If you want to know more….

- Come to
  - Benjamin's tutorial "Theory for Non-Theoreticians" (Sat. 10:40)
  - my tutorial "Non-Static Parameter Choices in Evolutionary Computation" (Sun. 14:00)
  - many other theory tutorials
  - THEORY track talks at GECCO
- Last but not least, talk to us ! (coffee breaks, evenings, after the sessions, by e-mail, etc.)