Black-Box Complexity:
From Complexity Theory to Playing Mastermind

Benjamin Doerr[1] & Carola Doerr[2]
[1]École Polytechnique, Paris-Saclay, France
[2]CNRS & Univ. Pierre et Marie Curie, Paris, France

http://www.sigevo.org/gecco-2014/

GECCO '14, Jul 12-16 2014, Vancouver, BC, Canada
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2605352

Instructors I/II

- Benjamin Doerr is a full professor at the French École Polytechnique. He also is a senior researcher at the Max Planck Institute for Informatics (Germany) and an adjunct professor at Saarland University.
- He received his diploma (1998), PhD (2000) and habilitation (2005) in mathematics from Kiel University. His research area is the theory both of problem-specific algorithms and of randomized search heuristics like evolutionary algorithms. Major contributions to the latter include runtime analyses for evolutionary algorithms and ant colony optimizers, as well as the further development of the drift analysis method, in particular, multiplicative and adaptive drift. In the young area of black-box complexity, he proved several of the current best bounds.
- Together with Frank Neumann and Ingo Wegener, Benjamin Doerr founded the theory track at GECCO, served as its co-chair 2007-2009 and serves again in 2014. He is a member of the editorial boards of "*Evolutionary Computation*", "*Natural Computing*", "*Theoretical Computer Science*" and "*Information Processing Letters*". Together with Anne Auger, he edited the book "*Theory of Randomized Search Heuristics*".

Instructors II/II

- Carola Doerr, née Winzen, is a permanent researcher with the CNRS and the Université Pierre et Marie Curie (Paris 6).
- She studied mathematics at Kiel University (Diploma in 2007) and computer science at the Max Planck Institute for Informatics and Saarland University (PhD in 2011). Her PhD studies were supported by a Google Europe Fellowship in Randomized Algorithms. From Dec. 2007 to Nov. 2009, Carola Doerr has worked as a business consultant for McKinsey & Company, mainly in the area of network optimization. She was a post-doc at the Université 7 in Paris and the Max Planck Institute for Informatics in Saarbrücken.
- Carola's main research interest is in the theory of randomized algorithms, both in the design of efficient algorithms as well as in randomized query complexities. She has published several papers about black-box complexities. She has contributed to the field of evolutionary computation also through results on the runtime analysis of evolutionary algorithms and drift analysis, as well as through the development of search heuristics for solving geometric discrepancy problems.

Objectives of the Tutorial

- This is a tutorial on black-box complexity. This is currently one of the very hot and active topics in the theory of randomized search heuristics.
- We shall try our best to…
  - tell you on an elementary level what black-box complexity is and how it shapes our understanding of randomized search heuristics
  - give an in-depth coverage of some of what happened in the last three years
  - show you why this also is a fun topic
- Don't hesitate to ask questions when they come up!
- Finally: We are happy to receive feed-back on this tutorial (email, coffee breaks, receptions, …)

Agenda

- Part 1: Introduction to black-box complexity (BBC)
  - Motivation: complexity theory for randomized search heuristics (RSH)
  - Definition of BBC
  - Four benefits
- Part 2: Tools and techniques (in the language of guessing games)
  - From black-box to guessing games
  - A general lower bound
  - How to play Mastermind
  - A new game
- Part 3: From BBC to new algorithms
- Summary, open problems, [appendix]

Timeline

2002
- Droste, Jansen, Tinnefeld, Wegener. **A new framework for the valuation of algorithms for black-box optimization.** FOGA

2006
- Droste, Jansen, Wegener. **Upper and lower bounds for randomized search heuristics in black-box optimization.** Theory Comput. Syst. 39

2009
- Anil, Wiegand. **Black-box search by elimination of fitness functions.** FOGA

2010
- Lehre, Witt. **Black-box search by unbiased variation.** GECCO

2011
- Doerr, Johannsen, Kötzing, Lehre, Wagner, Winzen. **Faster black-box algorithms through higher arity operators.** FOGA
- Doerr, Winzen. **Towards a complexity theory of randomized search heuristics: Ranking-based black-box complexity.** CSR
- Rowe, Vose. **Unbiased black box search algorithms.** GECCO
- Doerr, Kötzing, Lengler, Winzen. **Black-box complexities of combinatorial problems.** GECCO
- Doerr, Kötzing, Winzen. **Too fast unbiased black-box algorithms.** GECCO
- Doerr, Winzen. **Black-box complexity: breaking the O(n log n) barrier of LeadingOnes.** EA

2012
- Doerr, Winzen. **Playing Mastermind with constant-size memory.** STACS
- Doerr, Winzen. **Reducing the arity in unbiased black-box complexity**. GECCO

2013
- Doerr, Spöhel, Thomas, Winzen. **Playing Mastermind with many colors.** SODA
- Doerr, Doerr, Ebel. **Lessons from the black-box: Fast crossover-based genetic algorithms.** GECCO
- Afshani, Agrawal, Doerr, Doerr, Green Larsen, Mehlhorn. **The query complexity of finding a hidden permutation.** Munro-60

2014
- Doerr, Doerr, Kötzing. **Unbiased black-box complexities of Jump functions--how to cross large plateaus.** GECCO

Part 1: Intro to Black-Box Complexity

- Why a complexity theory for RSH?
  - Understand problem difficulty!
- How?
  - Black-box complexity!
- What can we do with that?
  - General lower bounds
  - understand the working principles of EAs
  - thorn in the flesh
- [Different notions of black-box complexity]

Why a Complexity Theory for RSH?

- Understand problem difficulty!
  - Randomized search heuristics (RSH) like evolutionary algorithms, genetic algorithms, ant colony optimization, simulated annealing, … are very successful for a variety of problems.
  - Little general advice which problems are suitable for such general methods
  - Solution: Complexity theory for RSH
- Take a similar successful route as classic CS!
  - Algorithmics: Design good algorithms and analyze their performance
  - Complexity theory: Show that certain things are just not possible
  - The interplay between the two areas provoked many cool results

Algorithms vs. Complexity Theory for RSHs – An Example

Algorithm Analysis: Prove how a certain algorithm solves a particular problem.

The (1+1) EA finds a minimum spanning tree with an expected number of $O(m^2 \log(m\, w_{max}))$ fitness evaluations.

Complexity Theory: What can the best possible algorithm for this problem do **or not**.

No RSH can solve the Needle problem in an expected number of less than $(2^n+1)/2$ fitness evaluations.

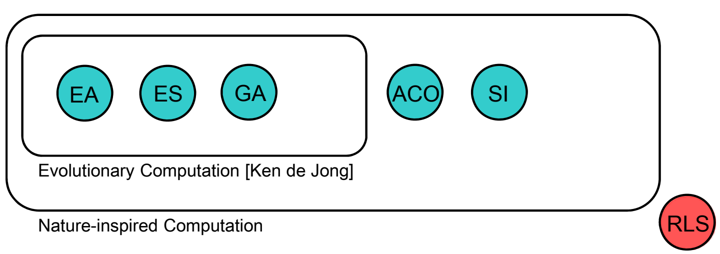
- Bottom line: Spanning tree is easy for RSHs, the Needle problem not.

Reminder: Classic Complexity Theory

- General approach: Complexity (difficulty) of a problem := Performance of the best algorithm on the hardest problem instance
- Example: "Sorting n numbers needs $\Theta(n \log(n))$ pair-wise comparisons."
  - Problem: "Sorting an array of n numbers"
  - Instance (input to algorithm): An (unsorted) array of n numbers
  - Algorithms: All that run on a Turing machine
  - Performance (cost) measure: Number of pair-wise comparisons
    - $T(A,I)$ = number of comparisons performed when algorithm A runs on instance I
  - Theorem: "Complexity of sorting = $\min_A \max_I T(A,I) = \Theta(n \log(n))$."
- How does this work for RSH?
  - Algorithms = RSHs, Performance = number of fitness evaluations, …

Complexity Theory for RSH

- Algorithms: Randomized search heuristics (RSH)
  - may generate solutions and query their fitness
  - no explicit access to the problem description
  - → *black-box optimization algorithm*



Complexity Theory for RSH

- Algorithms: Randomized search heuristics (RSH)
  - may generate solutions and query their fitness
  - no explicit access to the problem description
  - → *black-box optimization algorithm*
- Performance measure $T(A,I)$ = expected number of *fitness evaluations* until algorithm A running on instance I queries an optimum of I
- Black-box complexity: Expected number of fitness evaluations the best black-box algorithm needs to query the optimum of the hardest instance.
  - $\min_A \max_I T(A,I)$

'How many search point have to be evaluated to find the optimum."

BBC: What Can We Do With It?

- Black-box complexity: Expected number of fitness evaluations the best black-box algorithm needs to query the optimum of the hardest instance.
  - $\min_A \max_I T(A,I)$
- 4 benefits:
  - Measure for problem difficulty [that's how we designed the definition]
  - universal lower bounds
  - understand the working principles of EAs
  - a thorn in the flesh & a route to better algorithms

BBC: Universal Lower Bounds

- Black-box complexity: Expected number of fitness evaluations the best black-box algorithm needs to query the optimum of the hardest instance.
  - $\min_A \max_I T(A,I)$
- Follows right from the definition: The black-box complexity is a lower bound on the performance of any RSH!
  - BBC := $\min_A \max_I T(A,I) \le \max_I T(B,I)$ = performance of B
- Example:
  - Theorem [DJTW'02]: The black-box complexity of the needle function class is $(2^n+1)/2$.
  - Consequence: No RSH can solve the needle problem in sub-exponential time.
  - One simple proof replaces several proofs for particular RSH ☺

BBC: A Thorn in the Flesh

- If the black-box complexity is lower than what current best RSH achieve, you should wonder if there are better RSH for this problem!
- Example: OneMax functions
  - for all "bit-strings" $z \in \{0,1\}^n$ let f_z: $\{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto$ "number of positions in which x and z agree"
    - all f_z have a fitness landscape equivalent to the classic OneMax function (counting the number of ones in a bit-string).
  - Theorem: The black-box complexity of the class of all OneMax functions is $\Theta(n / \log(n))$.
  - But: All standard RSH need at least $\Omega(n \log(n))$ time!
  - Are there better natural RSH that we overlooked?

First answer: Part 3 of this tutorial

- Same motive as in classical theory: $n \times n$ matrix multiplication can be done in time $O(n^{2.3727})$, only lower bound is $\Omega(n^2)$.

BBC: Understand Working Principles

- Unbiased unary black-box complexity:
  $\min_A \max_I T(A,I)$,
  where as A we only regard unbiased algorithms using unary variation
  - unary (mutation-based): one parent gives one offspring
  - unbiased:
    - all bit-positions are treated equally
    - symmetry in the bit-values 0 and 1.
- Theorem [LW'10]: The unbiased unary BBC of OneMax is $\Omega(n \log n)$.
- "Insight": The reason for many simple RSH needing $\Omega(n \log n)$ iterations is that they are unbiased.
  - price for being unbiased is most $\Theta(\log n)^2$
  - fair price for having not relying on problem-specific knowledge ☺

Digression: Alternative BBC Models

- Previous slide:
  - restricted BBC models help understanding particular features of EA
  - different view: restricted BBC models might better capture the problem difficulty in evolutionary computation
- Next x slides: Discuss alternative black-box models
  - very active research area in the last 3 years
  - no definitive answer
- Common theme: Instead of allowing all black-box optimization algorithms, only regard a restricted class!
  - restricted class should include most classic RSH

begin digression

Alternative 1: Unbiased BBC

- Lehre&Witt (GECCO'10 theory track best paper award):
  - allow only unbiased variation operators: treat all bit-positions (1, …, n) and the two bit-values (0, 1) equally!
  - equivalent: if σ is an automorphism of the hypercube, then the probability that y is an offspring of $x_1, \dots, x_k$ must be equal to the probability that $\sigma(y)$ is an offspring of $\sigma(x_1), \dots \sigma(x_k)$
  - Observation: Most RSH are unbiased
    - exception: one-point crossover
  - Result: The unbiased, mutation-only BBC of OneMax is $\Theta(n \log(n))$
    - as observed for random local search, (1+1) EA, …
- Anti-result [DKW'11]: Also the TRAP$_k$ function has an unbiased, mutation-only BBC of $\Theta(n \log(n))$.
  - contrasts the $\Omega(n^k)$ performance of all classic RSH
- Interesting [DJKLW'11]: Unbiased 2-ary BBC of OneMax: $O(n)$.

Crossover helps?

Alternative 2: Ranking-Based BBC

- DW'11, following a suggestion by Niko Hansen (similar ideas in a paper by Olivier Teytaud): ranking-based
  - do not regard the absolute fitness values, but make all decisions dependent only on how fitnesses of search points compare!
  - Observation: Many RSH follow this scheme
    - exception: fitness-proportionate selection
  - Bad news: OneMax has a ranking-based BBC of $\Theta(n / \log(n))$ ☹
  - Good news: For BinaryValue…
    - BBC: $\log(n)$
    - ranking-based BBC: $\Omega(n)$
    - many RSH: $\Theta(n \log n)$
  - Open problem: Partition…
    - BBC: $O(n)$, heavily exploits absolute fitness values
    - Ranking-based: Maybe exponential?

Alternative 3: Memory-Restricted BBC

- Droste, Jansen, Wegener (Theor. Comput. Syst. 2006):
  - suggest to restrict the memory: store only a fixed number of search points and their fitness
  - inspired by bounded population size
  - conjecture: with memory one, the BBC of OneMax becomes the desired $\Theta(n \log(n))$
- DW'12: Disprove conjecture.
  - Even with memory one, the BBC of OneMax is $\Theta(n / \log(n))$.
    [I'll give some proof ideas in the second part of the tutorial]

Summary Alternative BBC Models

- Different models:
  - unrestricted (classic)
  - unbiased: don't exploit the encoding of solutions
  - ranking-based: only compare fitnesses
  - memory-restricted
- None is yet "the ultimate complexity notion" for RSH
  - Big open problem…
- Each expanded our understanding
  - what makes a problem hard
  - what makes a RSH powerful

end digression

Summary Part 1

- Black-box complexity (BBC): "Minimum number of search points that have to be evaluated to find the optimum"
  - Expected number of fitness evaluations the best black-box algorithm needs to query the optimum of the hardest instance.
  - $\min_A \max_I T(A,I)$
- Benefits:
  - Measure of problem difficulty
  - universal lower bounds
  - understand the working principles of EAs
  - thorn in the flesh & route to better algorithms

Part 2: Tools and Techniques

Plan for the 2nd part of this tutorial:

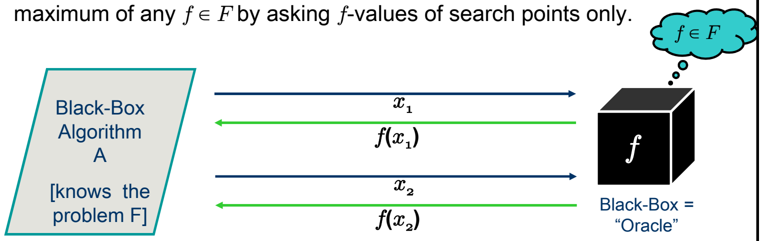
- Explain, why BBC and guessing games are almost the same
- Use the language of guessing games to demonstrate some techniques
  - Random guessing:
    - The BBC of OneMax or "how to play Mastermind with two colors?"
  - A simple "information theoretic" lower bound
  - Clever random guessing:
    - Mastermind with n colors
    - Memory-restricted BBC of OneMax = Mastermind with 2 rows
    - The LeadingOnes game

A Formal Definition of BBC

- Optimization problem: A set F of functions f: $\{0,1\}^n \to \mathbb{R}$
  - Aim is to find the maximum of a given $f \in F$.
  - Language:
    - An $f \in F$ is called an "instance of F"
    - $\{0,1\}^n$ "search space"
    - $x \in \{0,1\}^n$ "search point"
  - Example "Maximum Clique": For each graph G on the vertex set $\{1,\dots,n\}$, $f_G(x)$ is the size of the vertex set represented by x, if this is a clique in G, and 0 otherwise. $F := \{f_G \mid G$ a graph with vertices $1,\dots,n\}$.
- A black-box algorithm for F: A randomized algorithm that finds the maximum of any $f \in F$ by asking f-values of search points only (no explicit access to the instance f, e.g., the graph G in the clique example).

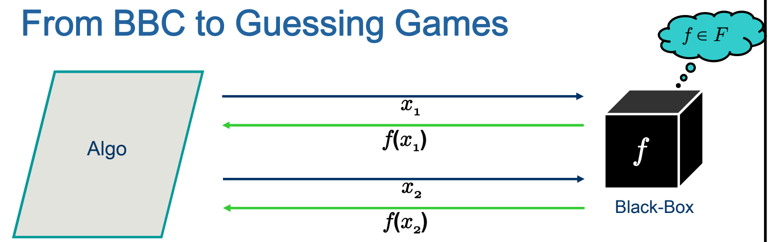
A Formal Definition of BBC

- A black-box algorithm for F: A randomized algorithm A that finds the maximum of any $f \in F$ by asking f-values of search points only.



- Performance $T(A,f)$ of A for $f \in F$: Expected time until an x with $f(x)$ = OPT(f) is queried
- Performance $T(A,F)$ of A on F: $\max_{f \in F} T(A,f)$
- BBC of F: $\min_A T(A,F)$, where A runs over all black-box algorithms for F

From BBC to Guessing Games



- Guessing game:
  - BlackBox chooses a hidden $f \in F$.
  - Algo tries to guess an x with $f(x)$ maximal
  - For each incorrect guess, BlackBox tells $f(x)$ to Algo
- Optimal strategy for Algo = optimal black-box algorithm
- Optimal strategy for black-box = "most difficult" $f \in F$
- Optimal number of rounds in the game = BBC(F)

Classic Guessing Game: Mastermind

- 2-player game
  - *CodeMaker* hides a ~~4~~ n-digit ~~6~~ k-color code C.
  - *CodeBreaker* tries to guess it using few guesses
- Guess: Some color code G
- Answer:
  - Number of positions in which C and G agree ("black answer-pegs" [here: red])
  - ~~Number of additional code letters that occur in a wrong position ("white pegs")~~

2-Color Mastermind = BBC(OneMax)

- OneMax test function: f: $\{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto$ "number of ones in x"
  - easy to find the unique global optimum (1,…,1).
  - RLS, (1+1) EA, … do this in $\Theta(n \log n)$ time.
- (Generalized) OneMax function, OneMax problem:
  - For each $z \in \{0,1\}^n$, let f_z: $\{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto$ "number of bits in which x and z agree"
  - All f_z have isomorphic fitness landscapes
  - OneMax problem: $F := \{f_z \mid z \in \{0,1\}^n\}$, the set of all OneMax functions
- Observation: Mastermind with the two "colors" 0 and 1 corresponds to the black-box complexity BBC(F)

Mastermind: 3 (?) Results

- $\Theta(n / \log n)$ guesses sufficient&necessary for $k = 2$ (BBC of OneMax)
  - Anil, Wiegand: "Black-box search by elimination of fitness functions". *Foundations of Genetic Algorithms* (FOGA) (2009)
  - lower bound from [DJW06]
- $\Theta(n \log k / \log n)$ for $k \leq n^{1-\varepsilon}$
  - Chvátal: "Mastermind". *Combinatorica* (1983)
- $\Theta(n / \log n)$ for $k = 2$
  - Erdős, Rényi: "On two problems in information theory". *Magyar Tud. Akad. Mat. Kutató Int. Közl* (1963)

Proof: Random Guessing

- CodeBreaker's strategy:
  - Guess $\Theta(n / \log n)$ random codes.
  - Look at all answers.
  - With high probability, no secret code other than the true one leads to these answers [elementary, straight-forward computation]
- Comments:
  - *Erdős probabilistic method* at its best.
  - Best possible (apart from constant factors hidden in $\Theta(\ldots)$)
  - Note: Non-adaptive strategy – questions do not depend on previous questions and answers.

A General Lower Bound

- [Droste, Jansen, Wegener'06] Consider a guessing game such that
  - there are s different secrets
  - each query has at most k different answers ($k \geq 2$).
  - Then the expected number Q of queries necessary to find the secret is at least $(\log_2(s) / \log_2(k)) - 1 = \log_k(s) - 1$.
- Information theoretic view: To encode the secret in binary, you need $\log_2(s)$ bits. Each answer can be encoded in $\log_2(k)$ bits. If Q rounds suffice, $Q \log_2(k)$ bits could encode the secret. [1)]
- Game-theoretic view: In the game tree, each node has at most k children. Hence at height Q, there are at most k^Q nodes. If s is bigger, then at some nodes, more secrets are possible. [1)]

[1)] Argument correct for deterministic strategies. For randomized ones, in addition, Yao's minimax principle is needed.

Back to 2-Color Mastermind…

- Lower bound: $(1 + o(1))\, n / \log_2(n)$
  - Argument: 2^n possible secrets, $n+1$ possible answers
    → general lower bound: $\log_2(2^n) / \log_2(n+1) = (1+o(1))n / \log_2(n)$
  - Information theoretic view: "learn at most $\log_2(n)$ bits per question"
- Upper bound computed precisely: $(2 + o(1))\, n / \log_2(n)$
  - Weaker by a factor of 2
  - Reason (informal): Typically, a random question yields an answer between $n/2 - \Theta(\sqrt{n})$ and $n/2 + \Theta(\sqrt{n})$.
    - "learn $\log_2(\Theta(\sqrt{n})) \approx (1/2) \log_2(n)$ bits per question"
    - game tree has relevant degree of only $\Theta(\sqrt{n})$.
- Big open problem (already mentioned in the Erdős-Rényi paper): What is the correct bound? Can you ask better questions?

Part 2: Tools and Techniques

Plan for the 2nd part of this tutorial:

- Explain, why BBC and guessing games are almost the same
- Use the language of guessing games to demonstrate some techniques
  - Random guessing:
    - The BBC of OneMax or "how to play Mastermind with two colors?"
  - A simple "information theoretic" lower bound
  - Clever random guessing: we are here now
    - Mastermind with n colors
    - Memory-restricted BBC of OneMax = Mastermind with 2 rows
    - The LeadingOnes game

Mastermind for $k = n$

- Best known lower bound: $\Omega(n)$
  - Information theory: n^n secrets, each query has ≤ n+1 answers
- Best known upper bounds: $O(n \log(n))$
  - Chvátal (Combinatorica'83): $2n \log(n) + 4n$
  - Chen, Cunha, Homer (COCOON'96): $2n \log(n) + 2n + 3$
  - Goodrich (IPL'09): $n \log(n) + 3n - 1$
  - [Random guessing takes $\Theta(n \log(n))$ guesses.]
- What is your guess?
  - [Problem open for 30 years, so no reason to be shy]

Plan: Clever Random Guessing

- Random guessing takes $\Theta(n \log(n))$ guesses.
- Informal justification:
  - The expected answer to a random question is 1.
  - "learn only a constant number of bits per question".
  - Information theory: $\log(n^n)/\log(\text{constant}) = n \log(n)$ questions
- Can we ask better questions?
  - Info-theory: We need to "learn more bits per question"
  - Problem: For the first question, the expected answer is 1, no matter what we ask (→ learn constant number of bits ☹)
  - If something works, it must be adaptive: Current question uses previous answers!

Reducing the Number of Colors

- Observation 1: Reducing the numbers of colors would help
  - Chvátal+: For $k \leq n$ colors, $O(n \log(k) / \log(n/k))$ random guesses suffice to find the secret code
    - Intuition: $s = k^n$, a random guess has an expected answer of $\lambda = n/k$, this and the $\approx \lambda^{1/2}$ adjancent answers show up with roughly equal probability → learn $\log(\lambda^{1/2}) = \Theta(\log(\lambda))$ bits
  - Note: For $k = n/\lambda$, this is $O(n \log(n) / \log(\lambda))$
    → any $\lambda = \omega(1)$ would improve ☺
- Observation 2: Reducing the number of colors is possible
  - for $k = n$ colors, the probability that a random guess gets a "0"-answer, is $(1 - (1/n))^n \approx 1/e \approx 0.37$
  - Such a 0-guess tell us, for each position, one color that cannot be there
    → essentially reduced the number of colors by one ☺

Details: Reducing the Number of Colors

- For k colors and n positions, the probability that a random guess is answered "0", is $(1 - (1/k))^n \geq 4^{-n/k}$.
- Reducing the number of colors from n to $4n$ / loglog(n) takes time at most $n\, 4^{-n/(4n / \log\log(n))} = n (\log n)^{1/2}$.
- Chvátal+: With only $k = 4n$ / loglog(n) colors possible at each position, random guessing needs $O(n \log(n) / \log\log(n))$ queries.
  - Intuition: "learn Θ(logloglog(n)) bits per query"
- First "Theorem": O(n log(n) / logloglog(n)) questions suffice!

Quickly Reducing the Colors

Reducing the number of colors from k to k-1:

- so far: get a "0"-answer after at most $4^{n/k}$ random guesses
- Example: $k = n/100$.
  - Random guess has an expected answer of 100.
  - Time to wait for a "0" is $(1+o(1))\, e^{100}$.
  - → Waiting for something quite rare ☹
- better: Partition the n positions into 100 blocks of equal size n/100 and ask randomly in each block (fill up the rest with dummy colors)
  - expected contribution per block: 1
  - waiting time for a "0" in a block: constant
  - total time: 100 times constant

Details: Quick Color Reduction

- Improved reducing the number of colors from k to k-1:
  - Assume that for each position we have a dummy color that for sure does not appear there [can, e.g., be found in $O(n)$ time]
  - Partition the n positions into n/k blocks of roughly equal size.
  - `For each block do`
    - `repeat`
      - `ask random colors in the block, put a dummy color in the rest`
    - `until answer = 0`
  - expected waiting time for "answer = 0": at most 4
  - total expected number of queries: at most 4 n/k [previously: $4^{n/k}$] ☺☺☺
- Total time to reduce the number of colors from k to k/2:
  - at most $(k/2)\, 4\, n / (k/2) = 4n$

Details (2): Quick Color Reduction

- Just proved:
  You can reduce the number of colors from k to k/2 colors in $4n$ queries
- Goodrich (2009): log(n) times halving the colors finds the secret code in $O(n \log n)$ questions [apart from constants, the same bound as Chvátal]
- DSTW'12: Reduce colors, then random guessing
  - Do the halving trick $\sqrt{\log n}$ times [$O(n \sqrt{\log n})$ queries]
    → $k = n / 2^{\sqrt{\log n}}$ colors possible at each position
  - Random guesses: $O(n \log(k) / \log(n/k)) = O(n \sqrt{\log n})$ random guesses using only these k colors find the secret
    → "learn $\log(2^{\sqrt{\log n}}) = \sqrt{\log n}$ bits per question"
- Theorem: Solve Mastermind with k=n colors in $O(n \sqrt{\log n})$ questions ☺

Ultra-Quick Color Reduction

- So far:
  - Reducing colors allows better queries ☺
  - from k to k-1 colors in n/k queries
    - color reduction queries: constant info gain (exp. answer = 1)
    - random query: expected answer n/k, info gain $\Theta(\log(n/k))$
    - Pay for asking "color reduction queries" ☹
- Solution [DSTW'13]: Ask questions that
  (i) reduce the number of colors, and
  (ii) tell us $\Theta(\log(n/k))$ bits of information on the secret.
  - "$k \to k$-1" in $O((n/k) / \log(n/k))$ queries instead of $O(n/k)$.
  - "$k \to k/2$" in $O(n / \log(n/k))$ queries instead of $O(n)$.
  - "$n \to n/2^i$" in $O(n \log(i))$ queries instead of $O(n\, i)$! [harmonic series]
  - i=log(n): Find the secret code in $O(n \log\log(n))$ queries ☺☺☺☺☺.

Here is How We Do This:

- Again:
  - Assume that we have a dummy color for each position.
  - k colors, n positions, $\lambda = n / k$ expected answer of random guess.
- Find a guess G with answer at most 2λ [expected constant time].
- Partition the positions into 4λ blocks of equal size
  → half of them contain no correct code letter ("empty block")
- Plan: Identify these with $\Theta(\lambda / \log \lambda)$ queries [next slide]
  → reduces the number of possible colors for $n/2$ positions
  - some Chernoff bounds: This is sufficient...

Indentifying Empty Blocks

- Situation:
  - a guess G with answer at most 2λ.
  - 4λ blocks, at least half of them empty.
- Query "dummy out random blocks": For each block independently do
  - with prob. ½: copy the block from G
  - with prob. ½: fill the block with dummy colors
- Analysis:
  - Expected answer: λ "learn $\Theta(\log \lambda)$ bits"
  - Some calculations: $\Theta(\lambda / \log \lambda)$ queries suffice to detect the empty blocks.
- Done ☺ More details: SODA'13 or `http://arxiv.org/abs/1207.0773`

Method: Clever Random Guessing

- Needed: Ask increasingly powerful queries (adaptive)
  - first query reveals only constant amount of info
- Generally good idea: randomized queries
  - "fooling the adversary": impossible to find a good secret for CodeMaker
- 3 increasingly powerful ways to mix *cleverness* and *randomness*
  - random queries composed of possible colors (and wait for "0")
  - random blocks, rest dummy colors: quicker to get a "0"
  - "dummy out random blocks": don't wait for a zero, but learn "zeros" from these more expressive queries
- Next: Two examples from true black-box complexity

A Second Example of "Clever Guessing"

- Problem: Memory-restricted BBC of OneMax
  - Memory-restriction: From one iteration to the next, the BB-algorithm may only store k search points together with their fitness.
  - Conjecture [Droste, Jansen, Wegener'06]: For k = 1, the BBC of OneMax is $\Theta(n \log n)$ as known from the (1+1) EA.
- Transfer to guessing games:
  - This BBC problem is equivalent to Mastermind with two rows only.
- Theorem [DW'12]: You can win 2-row Mastermind with $O(n / \log n)$ queries.
- Corollary: The memory-one BBC of OneMax is $\Theta(n / \log n)$.

Details: Two Rows Suffice!

- Result: On a board with two rows, you can still find the secret code with $O(n / \log n)$ guesses!

- Precise rules:
  - We start the game with an empty board
  - If there is an empty row, CodeBreaker can enter a guess, which will be answered by CodeMaker
  - If there is no empty row, CodeBreaker must empty one of the two rows and *forget the content*.
- Theorem: CodeBreaker has a strategy that
  - finds the secret code in $O(n / \log n)$ rounds
  - uses two rows only (all actions depend solely on these rows).

Fewer Rows: Proof Ideas

- Original Mastermind: Guess $\Theta(n / \log n)$ random codes. Store all guesses and answers on the board. Think.
  - Needs $\Theta(n / \log n)$ rows.
- 3 ingredients of our proof:
  - Find parts of the code: Determine $\Theta(n^\varepsilon)$ code letters with $\Theta(n^\varepsilon / \log n)$ relatively random guesses (ε constant)
    - Do this $n^{1-\varepsilon}$ times: find the code with $\Theta(n^\varepsilon / \log n)$ rows.
  - Determine such a part with *constant* number of rows
    - Do this $n^{1-\varepsilon}$ times: find the code with $\Theta(1)$ rows.
  - Do everything in *two* rows

Proof Idea (1): Find Parts of the Code

- Lemma:
  Let $B \subseteq [n]$, $|B| = n^\varepsilon$. "part"
  Let $G_1, G_2, \ldots$ be $\Theta(n^\varepsilon / \log n)$ guesses such that
    - G_i is random in positions in B
    - All G_i are equal in positions in $[n] \setminus B$
  Then with high probability these guesses and answers determine the secret code in B.
- Argument:
  - Basically, we play the game in B (and use the previous proof)
  - Only difficulty: The answers we get "are not for B only", but for the whole guess
    - Same deviation for all guesses
  - Some maths: Not a problem, guesses also determine deviation ☺

Proof Idea (2): Same with $O(1)$ Rows

- Plan: Simulate the previous slide in $O(1)$ rows
- Example: Find the first $L = \Theta(n^\varepsilon)$ code letters
  - $B_1 := L$ random letters.
  - Guess B_1 1…1 in row 1 and learn answer A_1.
  - Guess B_1 A_1 1….1 in row 2 and ignore answer
  - $B_2 := L$ random letters
  - Guess B_2 1…1 in row 1 and learn answer A_2
  - Guess B_1 A_1 B_2 A_2 1…1 in row 3 and ignore answer
  - …
- As before: $\Theta(L$ / log $L)$ guesses determine the code in the block
- New: The whole search history of $\Theta(L$ / log $L)$ guesses and answers can be stored in one row
- Needs 3 rows: "Old storage + new guess → new storage"

"A_1": Suitably encoded with O(log n) of letters

Proof Idea (3): Two Rows Only

- Difficulty:
  - To enter a new guess, one of the two rows must be emptied
  - You must store and guess in the same row
    - Problem: Storage influences CodeMaker's answers!
  - All control information must also be stored in this one row
    - what is the block I'm just optimizing?
    - what am I currently doing (guessing, storing, finding the unique solution, finding the last few letters in a different way…)
- Solution:
  - technical.
  - read the paper at STACS'12 or **`arxiv.org/abs/1110.3619`**.

Summary: Memory-BBC of OneMax

- Result: The complexity of Mastermind remains at $\Theta(n$ / log $n)$ guesses even if we allow only two rows.
  - Key proof argument: Clever guesses inspired by random guesses
- Open problems / future work:
  - Our proof works for any constant number of colors – what happens for larger numbers of colors?
  - constant factors: "what's hidden in the Θ(…)"
    - does a memory restriction lose us a constant factor?

Finally: A New Guessing Game

- So far: BBC is strongly related to guessing games
  - In particular: BBC(OneMax) ≈ Mastermind
  - Therefore: Use fun games to solve BBC problems
- Now [next few slides]: Use BBC problems to derive a fun games ☺
  - LeadingOnes Game

LeadingOnes Test Functions

- Classic test function:
  - LeadingOnes: $\{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto \max\{i \in \{0,\ldots,n\} \mid x_1 = \ldots = x_i = 1\}$
    - "how many bits counted from the left are one"
  - Unique optimum $(1,\ldots,1)$
  - "Harder than OneMax": Each non-optimal solution has only one superior Hamming neighbor
- LeadingOnes function *class* LO_n:
  - Let σ be a permutation of $\{1,\ldots,n\}$
  - Let $z \in \{0,1\}^n$ ("target string")
  - $f_{z\sigma} : \{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto \max\{i \in \{0,\ldots,n\} \mid x_{\sigma(1)} = z_{\sigma(1)}, \ldots, x_{\sigma(i)} = z_{\sigma(i)}\}$
    - "how many bits, counted in the order of σ, are as in z
  - same fitness landscape as LeadingOnes

The LeadingOnes Game

Transfer the BBC(LO_n) problem into a guessing game:

- CodeMaker: Picks a secret code z and a secret permutation σ
- Round:
  - CodeBreaker guesses a bit-string $x \in \{0,1\}^n$
  - CodeMaker's answer: $f_{z\sigma}(x)$ = "how many code letters in the order of σ are correct?"
- Main message of this slide: This is fun to play with n=5 or n=6!
  - try it during the next talks ;-)
- Next few slides: The theory is fun as well…

Black-Box Complexity of LeadingOnes

- Reminder: LO_n consists of all functions
  - $f_{z\sigma} : \{0,1\}^n \to \{0,\ldots,n\}$; $x \mapsto \max\{i \in \{0,\ldots,n\} \mid x_{\sigma(1)} = z_{\sigma(1)}, \ldots, x_{\sigma(i)} = z_{\sigma(i)}\}$
- Black-box complexity of LO_n, lower bound [Droste, Jansen, Wegener'06]
  - $\Omega(n)$, because you need $\Theta(n)$ fitness evaluations even if σ = id
- Black-box complexity of LO_n, upper bounds
  - $O(n^2)$, run-time of RLS, (1+1) EA, …
  - $O(n \log(n))$: determine "the next bit" with $\log(n)$ queries by simulating binary search (flip half of the potential bit positions…)
    - Information theoretic view:
      - "next bit"-position is a number in $\{1,\ldots,n\}$, coding length $\log(n)$
      - a typical query teaches you a constant amount of information (fitness increases by a small constant or not)

BBC(LeadingOnes), cool upper bounds

- DW'11: $O(n \log(n) / \log\log(n))$ is enough.
  - "learn average of $\log\log(n)$ bits per guess"
- AADLMW'13: $O(n \log\log(n))$ is enough, but also necessary
  - "learn avg. $\log(n)/\log\log(n)$ bits per guess"
  - first "really deep" lower bound proof on BBCs
  - `http://eccc.hpi-web.de/report/2012/087/`
- Next slide: Key argument of the $O(n \log(n) / \log\log(n))$ proof
  - how to learn more than constant information

Proving $O(n \log(n) / \log\log(n))$: Outline

- Assume that you have a solution x with $f_{z\sigma}(x) = k$ and you know which k bit-positions are responsible for this. Denote by I the remaining bit-positions. Let $L := \log(n)^{1/2}$
- Step 1: Use $L^2 = \log(n)$ iterations to find a y with $f_{z\sigma}(y) = k + L$
  - Flip the bits in I with probability $1/L$, accept if improvement
  - Note: We don't learn which L bit-positions lead to the improvement!!!
- Step 2: Use $\log(n)^{3/2} / \log\log(n)$ queries to determine the L bit-positions
  - In y, flip the I-bits with probability $1/L$. Do so $\log(n)^{3/2} / \log\log(n)$ times.
  - Look at all outcomes with fitness $k+j$ and find out bit number $k+j+1$.
  - With high probability, the $\log(n)^{3/2} / \log\log(n)$ samples suffice to learn all L bit-positions
- Step 1+2: $\log(n)^{3/2} / \log\log(n)$ fitness evaluations to gain $\log(n)^{1/2}$ bits…

Summary Techniques

- Black-Box Complexity ≈ guessing games
  - eases the language, increases the fun
- Information theoretic bound
  - BBC ≥ log(|SearchSpace|) / log(|fitness_values|)
- Random guessing
  - often: BBC ≤ log(|SearchSpace|) / log(|typical_answers|)
- Clever guessing: Increase the information gain!
  - Mastermind: Reduce number of colors → increase |typical_answers|
  - LeadingOnes: Don't learn "the next bit", but gain information on several bits in parallel
  - Memory-restricted & unbiased BBC: Coding techniques

Part 3: Learning from the Black-Box

- Reminder (part 1): If the black-box complexity is lower than what current best RSH achieve, you should wonder if there are better RSH for this problem!
- Example: OneMax
  - Black-Box Complexity: $\Theta\left(\frac{n}{\log n}\right)$
  - Standard EAs: $\Omega(n \log n)$

>> What does an optimal black-box algorithm do that EAs do not?

Learning from the Black-Box (cont.)

- What does an optimal black-box algorithm do that EAs do not?
  - $O\left(\frac{n}{\log n}\right)$ algorithm: samples uniformly at random
  - $O(n)$ algorithm: flip one bit in each step

1	1	1	1	1	1	1	1	1	5
0	1	1	1	1	1	1	1	1	4
1	0	1	1	1	1	1	1	1	6
				…					
1	0	1	1	0	1	1	0	1	9

Learning from the Black-Box (cont.)

- What does an optimal black-box algorithm do that the EAs do not?
  - $O\left(\frac{n}{\log n}\right)$ algorithm: samples uniformly at random
  - $O(n)$ algorithm: flip one bit in each step

>> Both algorithms learn also from search points of lower fitness

- EAs usually do not: search points of lower fitness are often...
  - ...discarded immediately (elitist selection)
  - ...kept only with the hope that after a while they can improve and lead to an optimal solution in a different area of attraction

A Class of New GAs

Exploit inferior search points in a simple manner: $(1 + (\lambda, \lambda))$ GA

1. Sample x u.a.r.
2. for t=1,2,3,... do

   Mutation phase:
   1. Sample ℓ from $\mathcal{B}(n, p)$;
   2. for $i = 1, \ldots, \lambda$ do: Sample x^i from x by flipping ℓ bits
   3. Choose x^i with $f(x^i) = \max\{f(x^1), \ldots, f(x^\lambda)\}$ u.a.r.

   Crossover phase:
   1. for $i = 1, \ldots, \lambda$ do: Sample y^i from x, x^i by crossover
   2. Choose y^i with $f(y^i) = \max\{f(y^1), \ldots, f(y^\lambda)\}$ u.a.r.

   Selection phase:
   1. if $f(y^i) \geq f(x)$ then $x \leftarrow y^i$

E.g., $p = k/n$ and $k > 2$
Created offspring typically inferior to parent x

Results for the $(1 + (\lambda, \lambda))$ GAs

- Best results for $\lambda = k = \Theta(\sqrt{\log n})$: runtime is $O(n\sqrt{\log n})$
  >> first time a GA with an asymptotic gain over standard EAs is proven for OneMax

- General bound: $O\left(\left(\frac{1}{k} + \frac{1}{\lambda}\right) n \log n + (k + \lambda)n\right)$
  >> improvement over "classic" $\Theta(n \log n)$ bound for quite a range of different values for k and λ

- Adaptive choice: For $\lambda = k = \max\left\{\frac{\mathrm{n}}{\mathrm{n}-\mathrm{f(x)}}, 2\right\}$: runtime is $O(n)$
  - self-adaptive choice (1/5th rule): works well in experiments

- More details: DW'13 (last year's GA track best paper award)

Final Summary ☺

- Black-box complexity: Expected number of fitness evaluations the best black-box algorithm needs to query the optimum of the hardest instance.
  - $\min_A \max_I T(A, I)$
  - Note: lower bound on the performance of any EA, ACO, …

- Strongly related to guessing games
  - BBC(OneMax) ≈ Mastermind
  - BBC(LeadingOnes) ≈ what you should play in the next tutorial ☺
    [download the game from http://www.mpi-inf.mpg.de/~winzen/LeadingOnesGame.html]

- Interplay between runtime analysis and BBC theory may lead to new algorithms
  - analogous to research in classic algorithms

Thanks!

Some Open Problems

- Mastermind, BBC(OneMax):
  - 2 colors: determine the leading constant [very difficult, posed already in the Erdős-Rényi paper]
  - n colors: Is our $O(n \text{ loglog } n)$ bound tight? [difficulty unclear, possibly easy and we just overlooked the right idea]
  - memory-restricted BBC:
    - say something on the leading constants [possibly easy]
    - say something for non-constant k [possibly easy, start with $k < n^{1-\varepsilon}$]
  - Mastermind with faulty answers? [possibly easy, so far only one result by Huang, Chen, Lin (2006)]

Some Open Problems

- Unbiased black-box complexity:
  - Lower bounds for the k-ary unbiased BBCs of OneMax, e.g., $\Omega(n)$ for k=2 [difficulty unclear, best upper bounds DW'12]
  - Improved bounds for the k-ary BBCs of LeadingOnes [best known results in FOGA'11, potentially ideas from the AADLMW-result can be used?]

- Ranking-based black-box complexity: Prove that the ranking-based BBC of partition is much higher than the unrestricted one [maybe very hard ☹]

- Memory-restricted black-box complexities: Give examples of problems having a higher BBC with memory restriction than without [my guess: should be easy and we were just unlucky that OneMax is not such an example]

Some Open Problems

- Other black-box models
  - Find a black-box model that capture most RSHs, but avoids some of the unrealistic low complexities of previous models.
  - Models for particular algorithms classes: ACO, EDA, EMO, …

- Black-box complexities for combinatorial optimization problems
  - Improve some of the non-tight bounds in DKLW'11 [since this is the first and only paper on this topic, at least some improvements shouldn't be too difficult]
  - Regard other CO problems than shortest paths and minimum spanning trees.

Appendix

1 Summary

In the following appendix, we survey the known black-box complexities of classic test functions in evolutionary computation. We tried our best to be exhaustive, so wherever lower and upper bounds do not match, we feel that it is an open problem to close this gap. In the tables below, we highlight some of these open problems which we find particularly interesting and try to grade their problem difficulty. Of course, what looks difficult now might look easy in the future, and what looks difficult for us might be easy for other researchers. Hence these subjective difficulty estimates should not be takes too serious. Still, they might be helpful, in particular, for younger researchers.

Our rating scheme is as follows:

- $***\,**$ Most likely a really difficult problem. Classic open problem in discrete mathematics. Several researchers from both the mathematics and the computer science community have addressed this problem. Should be worth an immediate PhD.
- $***$ Most likely a quite difficult problem. We know that this problem has been looked at by a number of researchers in the evolutionary computation community without success. Solving it would impress a number of people.
- $**$ Interesting problem that could be solvable with reasonable effort, though some understanding of non-trivial previous work will be needed. A progress here should easily make a good conference publication.
- $*$ Nice problem. We see a good chance that it can be solved without a broader background in black-box complexity theory. Possibly a good first problem to try when interested in this field. Results still publishable at good venues.
- (unrated) No rating simply means that we did not want to highlight this as one of the problems where we feel that progress is most urgent. It could still be an interesting problem and

Contents

1 Summary **3**

2 Classic Test Functions **4**
2.1 OneMax 5
2.2 Linear Functions 7
2.3 BinaryValue 8
2.4 LeadingOnes 9
2.5 Jump 10
2.6 Needle and Trap Functions 11
2.7 Polynomials and Monomials of Bounded Degree 12

3 Black-Box Complexities of Combinatorial Problems **13**
3.1 MaxClique 13
3.2 Partition 14
3.3 Minimum Spanning Trees 15
3.4 Single-Source Shortest Paths Problem . . . 17
3.4.1 Multi-Criteria Fitness Function . . . 18
3.4.2 Single-Criterion Fitness Function . . 19
3.5 Sorting 21

2.1 OneMax

The generalized OneMax function class is the one that consists of all functions $f_z : \{0,1\}^n \to [0..n], x \mapsto |\{i \in [n] \mid x_i = z_i\}|$, $z \in \{0,1\}^n$. The table below summarizes the known lower and upper bounds for the black-box complexities of this function class. Bounds given without reference follow trivially from identical bounds in stronger models, e.g., the $\Omega(n/\log n)$ lower bound for the memory-restricted black-box complexity follows directly from the same bound the unrestricted model.

Model	Lower Bound		Upper Bound		E. Diff.
unrestricted	$\Omega(n/\log n)$	info.-theo.	$O(n/\log n)$	[ER63, AW09]	
	$(1+o(1))(n/\log n)$	[ER63]	$(2+o(1))(n/\log n)$	[Lin64, Lin65, CM66]	$***\,**$
unbiased, arity 1	$\Omega(n\log n)$	[LW12]	$O(n\log n)$		
unbiased, arity $2 \le k \le \log n$	$\Omega(n/\log n)$		$O(n/k)$	[DW12c, DDE13]	$***$
r.b. unrestricted	$\Omega(n/\log n)$		$O(n/\log n)$	[DW14]	
r.b. unbiased, arity 1	$\Omega(n\log n)$		$O(n\log n)$	[Müh92] for (1+1) EA	
r.b. unbiased, arity $2 \le k \le n$	$\Omega(n/\log n)$		$O(n/\log k)$	[DW14]	$**$
(1+1) memory-restricted	$\Omega(n/\log n)$		$O(n/\log n)$	[DW12b]	

(E.Diff. abbreviates estimated problem difficulty; r.b. abbreviates ranking-based; info-theo. the information-theoretic bound [Yao77], cf. also [DJW06].)

It is a major open question to determine the correct bound for unrestricted algorithms. While it is known that all *non-adaptive* algorithms need $(2 \pm o(1))(n/\log n)$ queries to determine the target string z, it is not known whether faster adaptive query algorithms exist.

As discussed in the tutorial, determining the black-box complexity of the OneMax function class is equivalent to identifying optimal winning strategies for the Mastermind game with 2 colors and n positions, cf. also [DW12b]. We believe that the tools needed to determine the correct bound for the OneMax function class are the same as needed to compute the correct query complexity of the Mastermind game with n positions and $k = n$ colors. The recent $O(n \log\log n)$ bound for this game that we have discussed in the tutorial can be found in [DSTW13]. The best known lower bound for this game is the information-theoretic one, which is linear in n.

The lower bound of Lehre and Witt for 1-ary unbiased black-box algorithms holds for any pseudo-Boolean function with a unique global optimum.

2.2 Linear Functions

The linear function class contains for all $z \in \{0,1\}^n$ and for all $w \in \mathbb{R}^n$ the function $f_{z,w} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i \in [n], x_i = z_i} w_i$. The following table summarizes known upper and lower bounds for the black-box complexity this function class.

Model	Lower Bound		Upper Bound		E. Diff.
unrestricted	$\Omega(n/\log n)$	cf. OneMax	$n+1$		***
unbiased, arity 1	$\Omega(n \log n)$	cf. OneMax	$O(n \log n)$	[DJW02] for (1+1) EA	
unbiased, arity $k \geq 2$	$\Omega(n/\log n)$		$O(n)$		***
r.b. unrestricted	$\Omega(n/\log n)$		$n+1$	(folklore)	
r.b. unbiased, arity $k \geq 2$	$\Omega(n/\log n)$		$O(n)$	[DJK+11]	

(E.Diff. abbreviates estimated problem difficulty; r.b. abbreviates ranking-based)

One of the main challenges here is to determine the correct unrestricted black-box complexity. We conjecture a linear lower bound.

2.3 BinaryValue

The generalized BinaryValue function class contains for every $z \in \{0,1\}^n$ and every $\pi \in S_n$ the function $f_{z,\pi} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i \in [n], x_{\pi(i)} = z_{\pi(i)}} 2^i$. The following table summarizes known upper and lower bounds for the black-box complexity of BinaryValue.

Model	Lower Bound		Upper Bound	
unrestricted	$\lfloor \log_2 n \rfloor$	(folklore)	$\lceil \log_2 n \rceil + 2$	(folklore)
ranking-based unrestricted	$> n-1$	[DW14]	$n+1$	(folklore)

The black-box complexity of non-permutation-invariant function class BinaryValue* $:= \{f_{z,\mathrm{id}_{[n]}} \mid z \in \{0,1\}^n\}$ is $2 - 2^{-n}$, cf. e.g., [DJW06, Theorem 4].

2.4 LeadingOnes

The LeadingOnes function class is the class $\{f_{z,\pi} : \{0,1\}^n \to [0..n], x \mapsto \max\{i \in [0..n] \mid \forall j \leq i : z_{\pi(j)} = x_{\pi(j)}\} \mid z \in \{0,1\}^n, \pi \in S_n\}$. The following table summarizes known upper and lower bounds for the black-box complexity of LeadingOnes.

Model	Lower Bound		Upper Bound		E. Diff.
unrestricted	$\Omega(n \log\log n)$	[AAD+13]	$O(n \log\log n)$	[AAD+13]	
unbiased, arity 1	$\Omega(n^2)$	[LW12]	$O(n^2)$	[Rud97] for (1+1) EA	
unbiased, arity 2	$\Omega(n \log\log n)$		$O(n \log n)$	[DJK+11]	
unbiased, arity ≥ 3	$\Omega(n \log\log n)$		$O(n \log(n)/\log\log n)$	[DW12a]	**
r.b. unbiased, arity ≥ 3	$\Omega(n \log\log n)$		$O(n \log(n)/\log\log n)$	[DW12a]	**

(E.Diff. abbreviates estimated problem difficulty; r.b. abbreviates ranking-based)

The black-box complexity of the non-permutation-invariant function class LeadingOnes* $:= \{f_{z,\mathrm{id}_{[n]}} \mid z \in \{0,1\}^n\}$ is $\frac{n}{2} \pm o(n)$, see e.g., [DJW06, Theorem 6].

2.5 Jump

The class Jump$_\ell$, the class of all generalized Jump$_\ell$ functions with jump size ℓ, consists of all functions $\{f_z \mid z \in \{0,1\}^n\}$ with

$$f_z : \{0,1\}^n \to [0..n], x \mapsto \begin{cases} n, & \text{if } x = z; \\ |\{i \in [n] \mid x_i = z_i\}|, & \text{if } \ell < |\{i \in [n] \mid x_i = z_i\}| < n - \ell; \\ 0, & \text{otherwise} \end{cases}$$

Known lower bounds are: $\Omega(n \log n)$ for the unary unbiased black-box complexity of the generalized jump functions with arbitrary jump size ℓ [LW12]. For arbitrary arity and for the unrestricted black-box complexity the lower bound $\Omega(n/\log n)$ follows from information-theoretic considerations. The unrestricted and the unbiased black-box complexities of the extreme jump function (i.e., Jump$_\ell$ with $\ell = n/2 - 1$ is $\Omega(n)$, which can be verified by information-theoretic considerations [DDK14].

Known upper bounds for the unbiased black-box complexities are as follows:

Arity	Constant Jump $\ell = O(1)$		Short Jump $\ell = O(n^{1/2-\varepsilon})$		Long Jump $\ell = (1/2-\varepsilon)n$		Extreme Jump $\ell = n/2-1$	
$k = 1$	$\Theta(n \log n)$	[DKW11]	$\Theta(n \log n)$	[DDK14]	$O(n^2)$	[DDK14]	$O(n^{9/2})$	[DDK14]
$k = 2$	$O(n)$	[DKW11, DJK+11]	$O(n)$	[DDK14]	$O(n \log n)$	[DDK14]	$O(n \log n)$	[DDK14]
$3 \leq k \leq \log n$	$O(n/k)$	[DKW11, DW12c]	$O(n/k)$	[DDK14]	$O(n/k)$	[DDK14]	$\Theta(n)$	[DDK14]

Using the following Lemma, which is taken from [DDK14], it is not hard to see that new (better) upper bounds for the OneMax function class would immediately translate into better upper bounds for Jump$_\ell$ (see [DDK14] for details).

Lemma 1. *For all constants ε and c and all $\ell \in O(n^{1/2-\varepsilon})$, there is a unary unbiased subroutine s using $O(1)$ queries to Jump$_\ell$ such that, for all bit strings x, $s(x) = $ OneMax(x) with probability $1 - O(n^{-c})$.*

2.6 Needle and Trap Functions

We summarize known upper and lower bounds for the generalized NEEDLE and TRAP function classes. NEEDLE consists of all functions $\{f_z \mid z \in \{0,1\}^n\}$ with

$$f_z : \{0,1\}^n \to \{0,1\}, x \mapsto \begin{cases} 1, & \text{if } x = z, \\ 0, & \text{otherwise,} \end{cases}$$

and the function class TRAP contains for all $z \in \{0,1\}^n$ the function

$$f_z : \{0,1\}^n \to [0..2n], x \mapsto \begin{cases} |\{i \in [n] \mid x_i = 1\}|, & \text{if } x \neq z, \\ 2n, & \text{otherwise.} \end{cases}$$

Model	Lower Bound		Upper Bound		E. Diff.
unrestricted	$(2^n+1)/2$	[DJW06]	$(2^n+1)/2$	[DJW06]	
All models allowing random sampling			2^n	(folklore)	*

2.7 Polynomials and Monomials of Bounded Degree

Droste, Jansen, and Wegener regard in [DJW06] the black-box complexity of the class of monotone pseudo-Boolean bounded degree polynomials. This contains for all $\mathcal{A} \subseteq \{A \subseteq [n] \mid |A| \leq d\}$ and all $w_{\mathcal{A}} \in \mathbb{R}_{>0}^{|\mathcal{A}|}$ the polynomial $f_{\mathcal{A},w_{\mathcal{A}}} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{A \in \mathcal{A}} w_A \prod_{i \in A} x_i$; the parameter d is the *degree bound.*

It is shown [DJW06, Theorem 7] that the black-box complexity of this function class is bounded from below by $2^{d-1}+1/2$ and from above by $O(2^d \log n + n^2)$. The upper bound applies also to the (3+1) memory-restricted setting. The unary unbiased black-box complexity of this function class is at most $O(2^d(n/d)\log(1+n/d))$ by a result of Wegener and Witt [WW05, Theorem 4.2] for the Randomized Local Search (RLS) algorithm. This bound is tight for RLS [WW05, Theorem 5.1].

3 Black-Box Complexities of Combinatorial Problems

3.1 MaxClique

MAXCLIQUE is the problem of determining the size of a maximum clique in a graph. Droste, Jansen, and Wegener [DJW06] regard the following class of functions, and give a simple algorithm that needs at most $\binom{n}{2}+1$ queries to compute the size of a maximum clique.

$$\left\{ f_G : 2^{[n]} \to, V \mapsto \begin{cases} |V|, & \text{if } V \text{is a clique in } G \\ 0, & \text{otherwise} \end{cases} \middle| \; G \text{ is a graph on } n \text{ vertices} \right\},$$

where $2^{[n]} := \{A \mid A \subseteq [n]\}$ denotes the power set of $[n]$.

This simple example is often cited to show that there exist NP-hard problems with small polynomial black-box complexity. That this is not an artifact of the unrestricted black-box model, but applies also to the unary unbiased black-box model was shown in [DKW11] for the PARTITION problem.

3.2 Partition

In [DKW11] an NP-hard subclass of the PARTITION problem is considered, and it is shown that the unary unbiased black-box complexity of this class is $O(n \log n)$. In the following, we briefly present the class PARTITION$_{\neq}$.

Partition. Whereas the decision version of the PARTITION problem asks the question "Given a multiset $\mathcal{I}$ of positive integers ("weights"), is it possible to split the set into two disjoint subsets $\mathcal{I} = \mathcal{I}_0 \dot\cup \mathcal{I}_1$ such that $\sum_{w \in \mathcal{I}_0} w = \sum_{w \in \mathcal{I}_1} w$?", the optimization version asks for a partition $(\mathcal{I}_0, \mathcal{I}_1)$ of $\mathcal{I}$ such that the difference $|\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w|$ is minimized.

Partition$_{\neq}$. It is easily seen that PARTITION remains NP-hard if we restrict the problem to instances with all weights distinct. Let PARTITION$_{\neq}$ be the class of all instances $\mathcal{I}$ of PARTITION with $v \neq w$ for all $v, w \in \mathcal{I}$. Given an instance $\mathcal{I}$ of PARTITION$_{\neq}$, let us fix some enumeration $\sigma : \mathcal{I} \to [n]$ of the elements of $\mathcal{I}$. Let

$$f_{\mathcal{I}} : \{0,1\}^n \to \mathbb{Z}, x \mapsto \sum_{i \in [n], x_i = 0} \sigma^{-1}(i) - \sum_{i \in [n], x_i = 1} \sigma^{-1}(i).$$

The result in [DKW11] states that the unary unbiased black-box complexity of the function class $\{f_{\mathcal{I}} \mid \mathcal{I} \in \text{PARTITION}_{\neq}\}$ is $O(n \log n)$. Note here that we aim at minimizing the functions $|f_{\mathcal{I}}|$. The result also applies to the function class $\{|f_{\mathcal{I}}| \mid \mathcal{I} \in \text{PARTITION}_{\neq}\}$.

The unrestricted black-box complexity and the 3-ary unbiased black-box complexity of PARTITION is linear in the size $|\mathcal{I}|$.

3.3 Minimum Spanning Trees

It is one of the most interesting questions currently in the area of runtime analysis to determine the exact runtime of the (1+1) Evolutionary Algorithm (EA) on the minimum spanning tree (MST) problem. This question is open since Neumann and Wegener [NW07] proved an upper bound of $O(m^2 \log(nw_{\max}))$ fitness evaluations that are needed until the (1+1) EA finds an optimal MST. Here n is the number of vertices, m the number of edges and $w_{\max}$ is the maximum of the positive and integral edge weights. It is widely believed that the dependence on the maximum edge weight is not necessary. However, so far this could be proven only for a randomized local search (RLS) variant doing one-bit and two-bit flips each with probability $1/2$, cf. [RS07].

The black-box complexity of MST has been analyzed in [DKLW13]. Since the MST problem has a natural representation via bit-strings, all existing black-box notions can be analyzed without further discussion. The only minor detail to take care of is that in the MST problem usually the fitness is a two-criteria one, that is, the fitness function returns both the number of connected components and the total weight of the solution. For all black-box notions apart from the ranking-based one, this provides no difficulties. For the ranking-based black-box complexity, the model in which the ranking information is given for each component of the fitness separately is regarded in [DKLW13]. All bounds except for the ones in the ranking-based model apply also to the MST model in which the single-criterion fitness function is used that penalizes each connected component by some large value $C > n^2 w_{\max}$.

The bounds from [DKLW13] are summarized in the following table. The unary unbiased black-box complexity reduces to $O(mn \log(m/n))$ if the edge weights are pairwise different.

Model	Lower Bound	Upper Bound	Estimated Difficulty
(ranking-based) unrestricted	$n-2$	$2m+1$	**
unbiased, arity 1	$\Omega(m \log n)$	$O(mn \log n)$	
ranking-based unbiased, arity 1	$\Omega(m \log n)$	$O(mn \log n)$	
(ranking-based) unbiased, arity 2	$\Omega(m / \log n)$	$O(m \log n)$	
(ranking-based) unbiased, arity 3	$\Omega(m / \log n)$	$O(m)$	**

3.4 Single-Source Shortest Paths Problem

Another intensively studied problem in the runtime analysis community is the single-source shortest paths (SSSP) problem. For a given graph $G = (V, E)$ with edge weights and a distinguished *source vertex* $s \in V$, the SSSP problem asks to determine for each vertex $w \in V \setminus \{s\}$ the shortest path between w and the source s, i.e., a path p_w that minimizes $\sum_{e \in p_w} w(e)$. For the SSSP problem, a bit-string representation of the solution candidates is not very natural. Therefore, [STW04] and all subsequent works represent individuals by (directed) shortest-paths trees.

3.4.1 Multi-Criteria Fitness Function

In this model, which is regarded in [DJW06], an algorithm may query arbitrary trees on V and the objective value of any such tree is an $n-1$ tuple of the distances of the $n-1$ non-source vertices to the source s (if an edge is traversed which does not exist in the input graph, the entry of the tuple is ∞ or some artificially large value). Known bounds for the black-box complexity of the SSSP problem in this setting are summarized in the following table. As argued in [DKLW13, Section 5.1], imposing certain symmetry conditions among the vertices makes little sense if the fitness function explicitly distinguishes them. Unbiased black-box complexities have therefore not been considered in the multi-criteria setting.

Model	Graph	Lower Bound		Upper Bound	
unrestricted	arbitrary	$n-1$	[DKLW13]	$n-1$	[DKLW13]
	complete	$n/4$	[DKLW13]	$\lfloor (n+1)/2 \rfloor + 1$	[DKLW13]
(2+1) memory-restricted	arbitrary			$2n-3$	[DJW06]

3.4.2 Single-Criterion Fitness Function

One may also consider the SSSP problem with a single-criterion fitness function, which assigns to each search point the sum of the distances of all vertices to the source. In this model, it is important that each unconnected vertex contributes some fixed large value to the fitness but not ∞. The solution candidates (search points) in this model are vectors $(\rho(v))_{v \in V \setminus \{s\}} \in V^{n-1}$ to be interpreted that the predecessor of a vertex $v \in V \setminus \{s\}$ is node $\rho(v)$. It is known that the running time of the (1+1) EA for this problem is $O(mn \log(nw_{\max}))$, see [DJ10]. Again, it is a well-known open problem whether the dependence on $w_{\max}$ is necessary or not.

For the single-criterion fitness function, it is an interesting question how to model unbiasedness. Three different versions are discussed in [DKLW13]: the *generalized unbiased* black-box complexity model as defined in [RV11], the *redirecting* unbiased model, and the *structure preserving* unbiased model.

Intuitively speaking, in the *structure preserving* unbiased black-box model, the operators do not regard the *labels* of different nodes, but only their structure. In the *redirecting* unbiased black-box model, intuitively, a node may choose to change its predecessor in the shortest path tree but if it decides to do so, then all possible predecessors must be equally likely to be chosen. In contrast to the structure preserving and the generalized unbiased black-box models, this notion seems to be much better suited for the SSSP problem. The bounds from [DKLW13] are summarized in the following table. The upper bounds for the generalized and the structure preserving unbiased black-box models differ from the unrestricted ones by at most one query. The lower bound for the redirecting unbiased model holds for arbitrary arity.

The lower bound for the unrestricted black-box complexity follows from the one for linear functions.

Model	Lower Bound	Upper Bound	Estimated Difficulty
unrestricted	$\Omega(n/\log n)$	$n(n-1)/2$	* * *
ranking-based unrestricted	$\Omega(n/\log n)$	$(n-1)^2$	
ranking-based redirecting unbiased, arity 1	$\Omega(n/\log n)$	$O(n^3)$	
redirecting unbiased, arity 2	$\Omega(n^2)$	$O(n^2 \log n)$	

3.5 Sorting

Bounds for the unrestricted and memory-restricted black-box complexities of different sorting problems can be found in [DJW06, Section 5].

References

[AAD+13] Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. The query complexity of finding a hidden permutation. In *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2013. Full version available online at http://eccc.hpi-web.de/report/2012/087/#revision1.

[AW09] Gautham Anil and R. Paul Wiegand. Black-box search by elimination of fitness functions. In *Proc. of Foundations of Genetic Algorithms (FOGA'09)*, pages 67–78. ACM, 2009.

[CM66] David G. Cantor and William H. Mills. Determination of a subset from certain combinatorial properties. *Canadian Journal of Mathematics*, 18:42–48, 1966.

[DDE13] Benjamin Doerr, Carola Doerr, and Franziska Ebel. Lessons from the black-box: Fast crossover-based genetic algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 781–788. ACM, 2013.

[DDK14] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Unbiased black-box complexities of jump functions—how to cross large plateaus. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'14)*. ACM, 2014. To appear.

[DJ10] Benjamin Doerr and Daniel Johannsen. Edge-based representation beats vertex-based representation in shortest path problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'10)*, pages 758–766. ACM, 2010.

[DJK+11] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Per Kristian Lehre, Markus Wagner, and Carola Winzen. Faster black-box algorithms through higher arity operators. In *Proc. of Foundations of Genetic Algorithms (FOGA'11)*, pages 163–172. ACM, 2011.

[DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[DJW06] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39:525–544, 2006.

[DKLW13] Benjamin Doerr, Timo Kötzing, Johannes Lengler, and Carola Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[DKW11] Benjamin Doerr, Timo Kötzing, and Carola Winzen. Too fast unbiased black-box algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*, pages 2043–2050. ACM, 2011.

[DSTW13] Benjamin Doerr, Reto Spöhel, Henning Thomas, and Carola Winzen. Playing Mastermind with many colors. In *Proc. of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 695–704. SIAM, 2013. Full version available at http://arxiv.org/abs/1207.0773.

[DW12a] Benjamin Doerr and Carola Winzen. Black-box complexity: Breaking the $O(n \log n)$ barrier of LeadingOnes. In *Proc. of Artificial Evolution (EA'11)*, volume 7401 of *LNCS*, pages 205–216. Springer, 2012.

[DW12b] Benjamin Doerr and Carola Winzen. Playing Mastermind with constant-size memory. In *Proc. of Symposion on Theoretical Aspects of Computer Science (STACS'12)*, pages 441–452. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[DW12c] Benjamin Doerr and Carola Winzen. Reducing the arity in unbiased black-box complexity. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'12)*, pages 1309–1316. ACM, 2012.

[DW14] Benjamin Doerr and Carola Winzen. Ranking-based black-box complexity. *Algorithmica*, 68(3):571–609, 2014.

[ER63] Paul Erdős and Alfréd Rényi. On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei*, 8:229–243, 1963.

[Lin64] Bernt Lindström. On a combinatory detection problem I. *Publications of the Mathematical Institute of the Hungarian Academy of Science*, 9:195–207, 1964.

[Lin65] Bernt Lindström. On a combinatorial problem in number theory. *Canadian Mathematical Bulletin*, 8:477–490, 1965.

[LW12] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[Müh92] Heinz Mühlenbein. How genetic algorithms really work. Mutation and hillclimbing. In *Proc. of Parallel Problem Solving From Nature (PPSN'92)*, pages 15–25. North-Holland, 1992.

[NW07] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378:32–40, 2007.

[RS07] Joachim Reichel and Martin Skutella. Evolutionary algorithms and matroid optimization problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 947–954. ACM, 2007.

[Rud97] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms.* Kovac, 1997.

[RV11] Jonathan Rowe and Michael Vose. Unbiased black box search algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*, pages 2035–2042. ACM, 2011.

[STW04] Jens Scharnow, Karsten Tinnefeld, and Ingo Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3:349–366, 2004.

[WW05] Ingo Wegener and Carsten Witt. On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability & Computing*, 14:225–247, 2005.

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. of Foundations of Computer Science (FOCS'77)*, pages 222–227. IEEE, 1977.