Theory of Online Parameter Selection

[Some Thoughts, Many Questions]

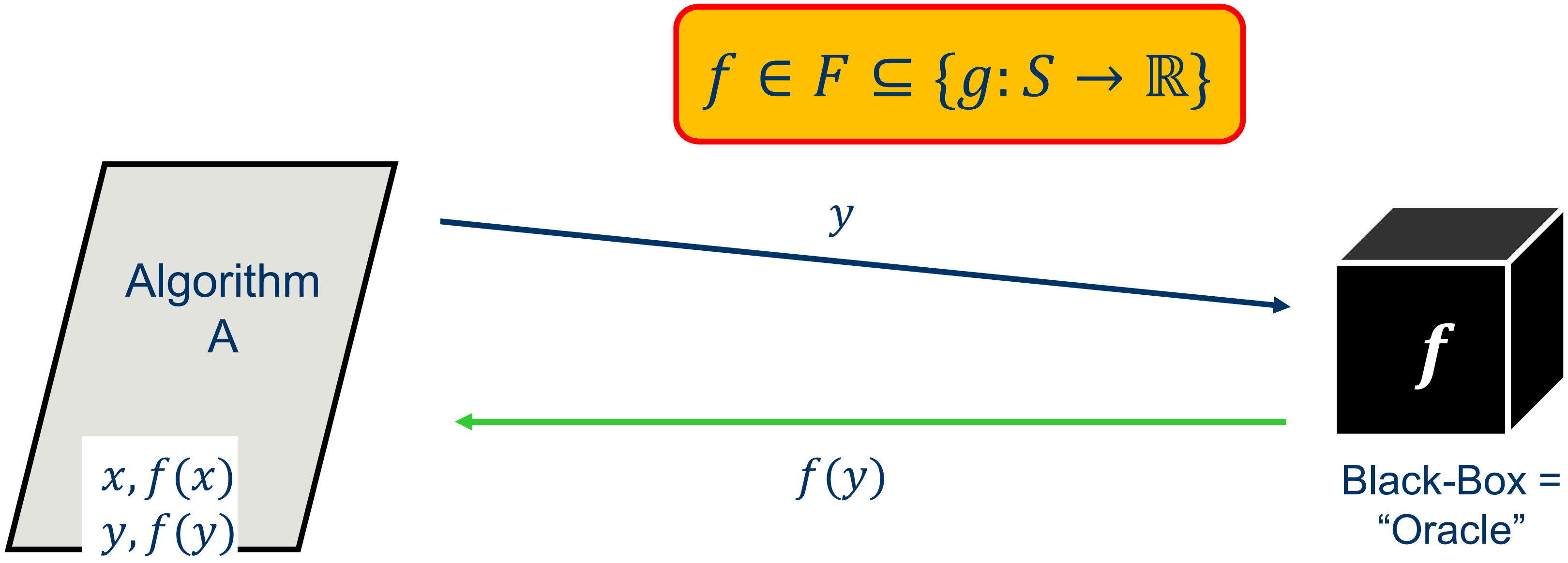
Carola Doerr

CNRS and Pierre et Marie Curie University, Paris, France

COSEAL Workshop

Brussels, September 11, 2017

Black-Box Framework



- **<u>Performance measure:</u>** number of function evaluations ("black-box *queries"/"oracle calls"*) needed (on average) to find solutions of certain quality

Online Parameter Selection

- Assume: fix problem, fix algorithm (←typically parametrized)
- Question: which parameter setting?

  → can have decisive impact on performance

  → hard to answer because of complex interactions between the parameters
- 1st answer: Tuning!
- 2nd (better?) answer: Online Parameter Selection (“*Parameter Control*”)

  → Hope: performance gains through automated adjustment of parameter setting
    - to the problem instance
    - to the state of the optimization process

  → Problem: ***how*** *to select parameters online?*

Is there room to discuss this question at COSEAL?

Online Parameter Selection

- Assume: fix problem, fix algorithm (←typically parametrized)
- Question: which parameter setting?

  → can have decisive impact on performance

  → hard to answer because of complex interactions between the parameters
- 1st answer: Tuning!
- 2nd (better?) answer: Online Parameter Selection (“*Parameter Control*”)

  → Hope: performance gains through automated adjustment of parameter setting

    - to the problem instance
    - to the state of the optimization process

  → Problem: ***how*** *to select parameters online?*

  → Has become a “hot topic” in randomized black-box optimization (but apparently also in ML)

  → My interest: theoretical foundations for online parameter selection (in discrete search spaces)

Theoretical Approach – Why?

- **Selfish (?) motivation:**
  - Mathematical curiosity ☺
  - Fun ☺
- **Hope for long-lasting impact:**
  - performance **guarantees** vs. empirical *observations*
    ($n^{1.2}$ algo. looks better than a $100n$ one for a loooooong time*!)
  - upper bounds: universality of results,
    e.g., performance guarantees for *any* linear/monotone/… function
  - lower bounds: what is the best possible performance that *any algorithm* can have?
  - understand **working principles** behind the processes
  - theory as **source of inspiration**

Happy to discuss this!

*if and only if $n \leq 10^{10}$

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…)

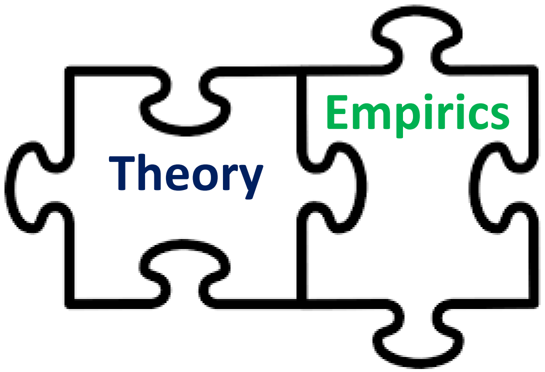
Theory and Experiments: *Complementary Results*

Theory

- cover all problem instances of arbitrary si…
  → guaran…
- proof tells…
- only mode…
  instances…
- limited sc…
- limited pre…
- implementation independent
- finding proofs can be difficult

Empirics

- only a finite number of instances
  …is is
  …ers
  …s
- …implement
- depends on implementation
- can be cheap (well, depends…)



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

Some Recent Results: 1. Local Update Rules

- Examples: provable performance gain for the (1+($\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$)) GA:
  - 3 parameters:
    - "*offspring population size*" λ (# points sampled per iteration)
    - "*mutation strength*" p (radius of the search)
    - *"crossover probability"* c (trading old vs. new information)

    → complex interactions between these parameters
  - mathematical proof: best static parameter choice gives a total expected runtime of $\Theta\left(n\frac{\sqrt{\log n \log\log\log n}}{\sqrt{\log\log n}}\right)$ on the problem of minimizing the Hamming distance [DoerrDoerr15, Doerr16, DoerrDoerr17]
  - Surprise: very simple online parameter selection mechanism yields a $\Theta(n)$ expected runtime [DoerrD15b,DoerrDoerr17]
    - <u>This is optimal!</u>

z = 1 1 1 1 1 0 0 0 0

x = 0 0 0 1 1 1 1 0 0

$f_z(x)$ = 5

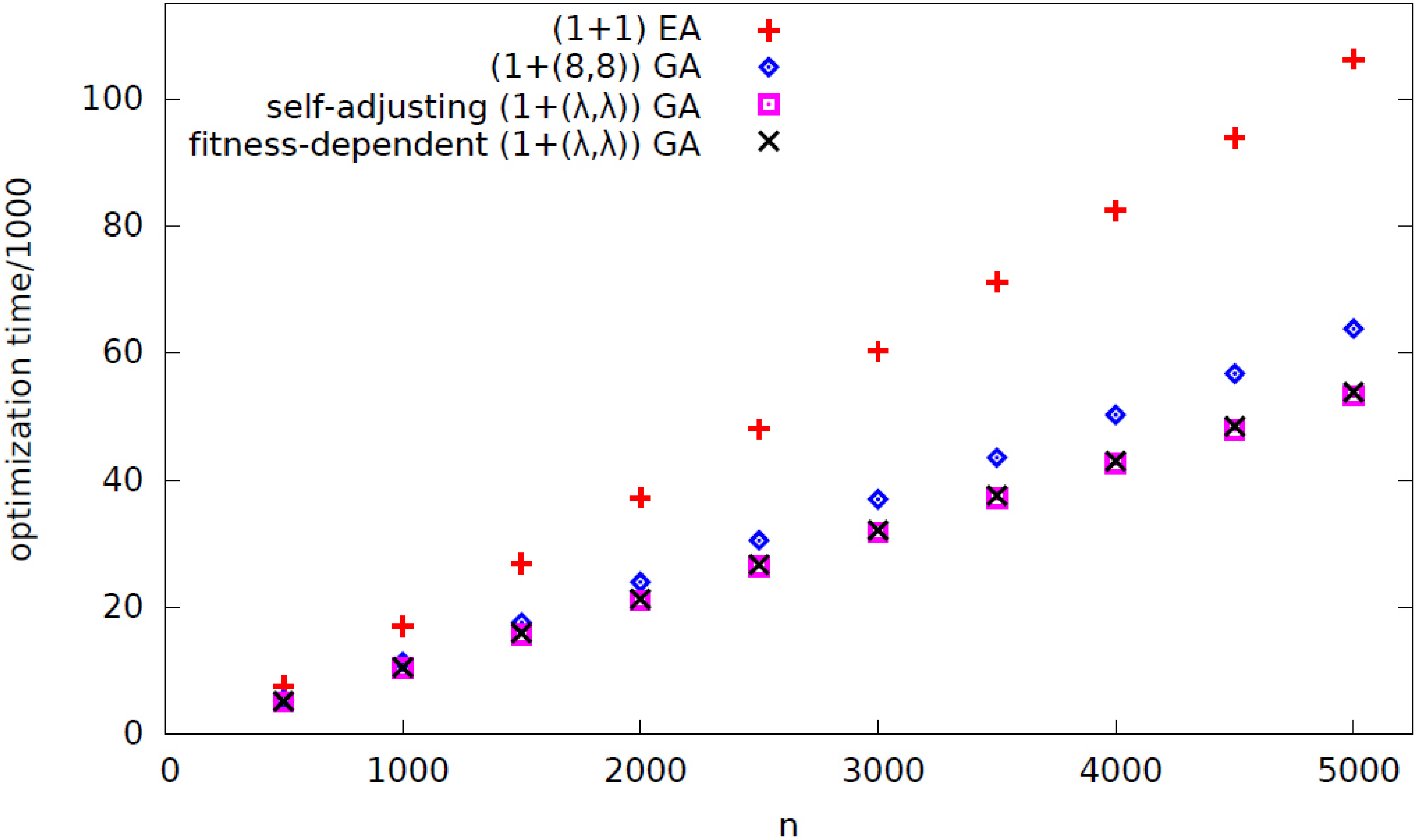
Simple Local Update Rule

- Optimal dynamic parameter choice
    - *"offspring population size"* $\lambda = \sqrt{\frac{n}{n-f(x)}}$
    - *"mutation strength"* $\boxed{p = \lambda/n}$
    - *"crossover probability"* $\boxed{c = 1/\lambda}$

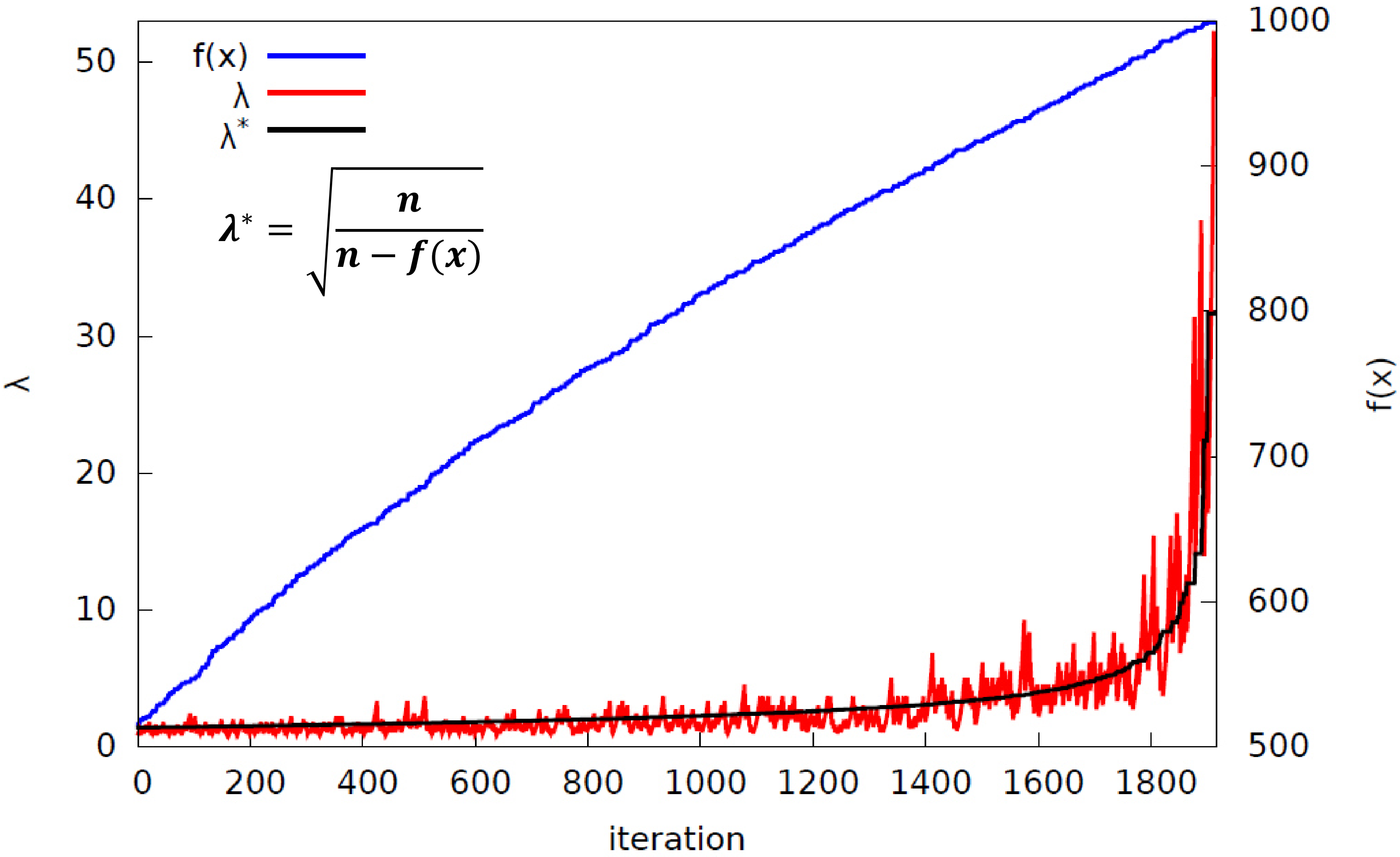
  → only one parameter left!
- 1/5th success rule:
  - If at the end of an iteration
    - we have an improvement ($f(y) > f(x)$) then $\lambda \leftarrow \lambda/F$;
    - No improvement ($f(y) \leq f(x)$) then $\lambda \leftarrow \lambda F^{1/4}$;

Optimal parameter selection schemes can be very simple!

Experimental Results for Self-Adjusting Version

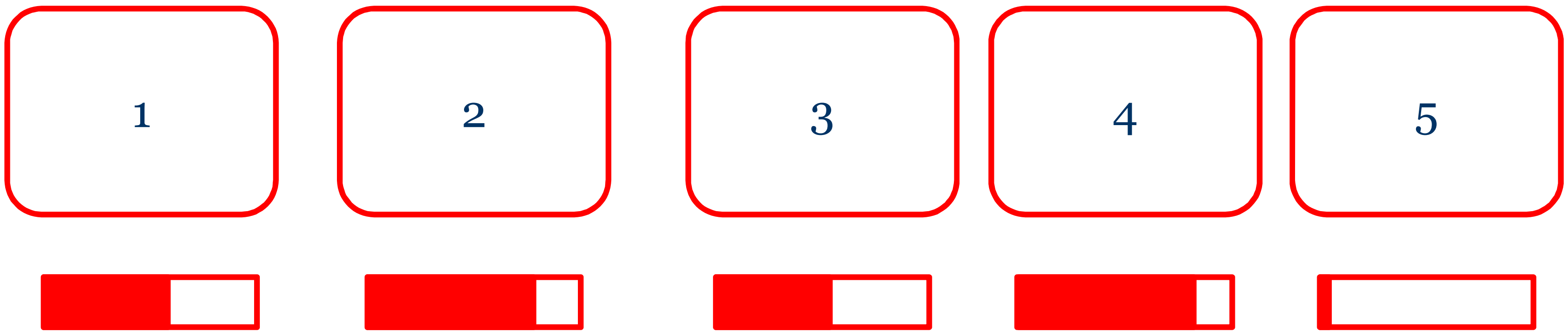


Example Run Self-Adjusting $(1 + (\lambda, \lambda))$ GA



2^{nd} Example: Online *Learning*

- The main idea for learning-/reward-type adjustment rules is
  - have a set S of possible parameter values
  - according to some rule, test all or some of these values
  - update the likelihood to employ the tested value based on the feedback from the optimization process
- Picture to have in mind: multi-armed bandits (MAB)
  - K experts
  - in each round, you have to chose one of them and you follow his advice
  - you update your confidence in this expert depending on the quality of his forecast



2^{nd} Example: Online *Learning, Comments*

- **Exploitation vs. exploration trade off**
  - exploitation: we want, of course, to use an optimal parameter value as often as possible
  - exploration: we want to test each parameter value sufficiently often, to make sure that we select the "optimal" one
- **Differences to classical learning/ML setting**
  1. regret minimization (learning) vs. optimization
  2. "rewards" change over time! ($\neq$ "classical" MAB setting)
     → Frequently found feature: *time-discounted methods*. That is, a good advice in the past is worth less than a good advice now

UCB = Upper Confidence Bound

- **Upper Confidence Bound**, aka *UCB*-mechanisms are well known in learning theory, cf. work by Auer, Cesa-Bianchi, Fischer ML'02 [ACBF02]
- Main ideas:
  - cUCB greedily selects the operator (the "arm") maximizing the following expression:

$$\text{expected reward} + \sqrt{c \log \frac{\sum_k n_{k,t}}{n_{j,t}}},$$

    where
    - $n_{k,t}$ is the number of times the k-th arm has been pulled in the first t iterations and
    - c is a parameter that allows to control the exploration likelihood (vs. exploitation, which is controlled by the first summand)
  - tuned and other variants of this algorithm exist, cf. [ACBF02] for details and empirical evaluations

UCB = Upper Confidence Bound

- **Upper Confidence Bound**, aka *UCB*-mechanisms are well known in learning theory, cf. work by Auer, Cesa-Bianchi, Fischer ML'02 [ACBF02]
- Main ideas:
  - cUCB greedily selects the operator (the "arm") maximizing the following expression:

$$\text{expected reward} + \sqrt{c \log \frac{\sum_k n_{k,t}}{n_{j,t}}},$$

    where
    - $n_{k,t}$ is the number of times the k pulled in the first t iterations and
    - c is a para to control the exploration likelihood (vs. is controlled by the first summand)
  - other variants of this algorithm exist, cf. [ACBF02] for details and empirical evaluations

Mathematical performance guarantees?

(Almost) Optimality of Learning-Based Parameter Selection

- ε-greedy variable size neighborhood heuristic
  - Fix a small number of possible parameter values $[r] \coloneqq \{1, 2, \ldots, r\}$
  - Estimate the expected fitness gain $v_{t-1}[k]$ from using k-bit flips (using data from the past, see next slide)
  - In iteration t
    - with probability δ, use a random $k \in [r]$ "*exploring* mut. strengths"
    - with prob. $1 - \delta$, use a k that maximized $v_{t-1}[k]$ "take the most efficient k"
  - Update the expected fitness gain estimations
- For the Hamming distance problem, this self-adjusting mutation strength in almost all iterations uses the (in this situation) optimal mutation strength.
- The iterations that do not operate with the optimal mutation rate account for an additive $o(n)$ contribution to the total runtime and are thus negligible
- This adaptive mechanism is provably faster than all static unbiased mutation operators!
- Fixed budget performance: our algorithm with the same budget computes a solution that asymptotically is 13% closer to the optimum than RLS (given that the budget is at least $0.2675n$).
- Promising empirical results for other problems

[Doerr, Doerr, Yang: PPSN 2016]

Estimating the Expected Fitness Gain

- Expected fitness gain estimation for using a k-bit flip:

$$v_t[k] := \frac{\sum_{s=1}^{t} 1_{r_s=k} \, (1-\varepsilon)^{t-s}\big(f(x_s) - f(x_{s-1})\big)}{\sum_{s=1}^{t} 1_{r_s=k} \, (1-\varepsilon)^{t-s}}$$

- $1/\varepsilon$: "*forgetting rate*", determines the decrease of the importance of older information. $1/\varepsilon$ is (roughly) the information half-life
- The "velocity" can be computed iteratively in constant time by introducing a new parameter $w_t[r] := \sum_{s=1}^{t} \mathbf{1}_{r_s=r}(1-\epsilon)^{t-s}$
- This mechanism seems to work well also for other problems
  - So far, no other theoretical results available
  - A few experimental results for LeadingOnes and the Minimum Spanning Tree problem exist, see next 2 slides (these results were also presented in [DDY16a])
  - Again, much more work is needed to see how the algorithm performs on other problems and how to set the parameters δ and ε

Questions

1. Is online parameter selection interesting for you?
   - What in particular? Or why not?
2. Research on dynamic multi-armed bandits in (M)L
   - state-of-the-art?
   - theoretical results?
3. (Poster session/Dinner/Coffee breaks/E-Mail/Paris:)
   Theoretical results in Algorithm Selection/Configuration
   - what is known?
   - what would be an interesting result for you?