Simple On-the-Fly Parameter Selection

Carola Doerr
CNRS and Sorbonne University, Paris, France
Markus Wagner
University of Adelaide, Australia

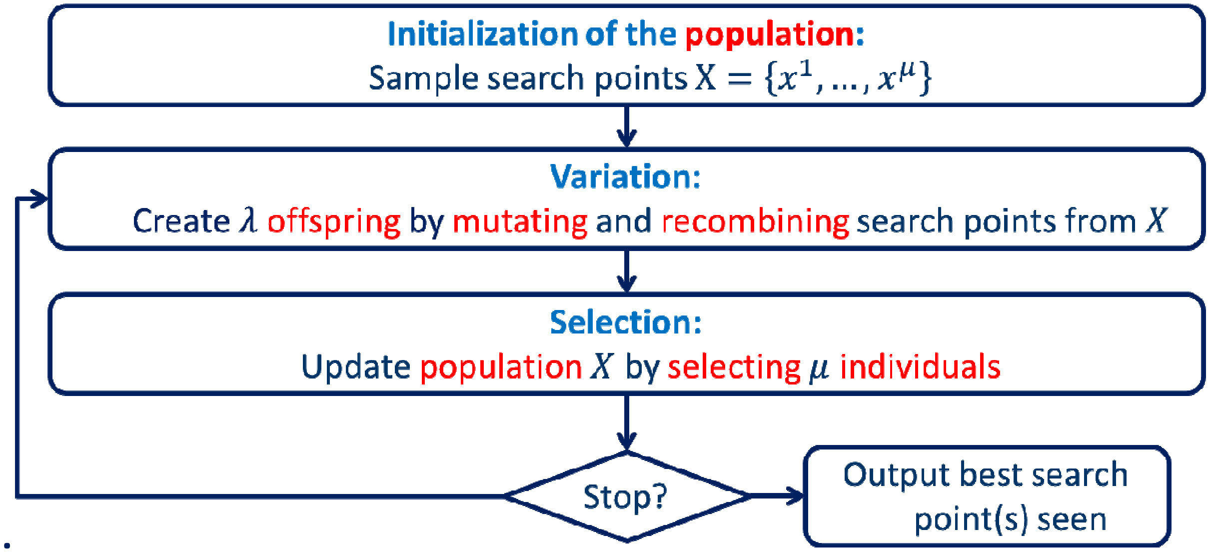
GECCO

Presentation at GECCO 2018

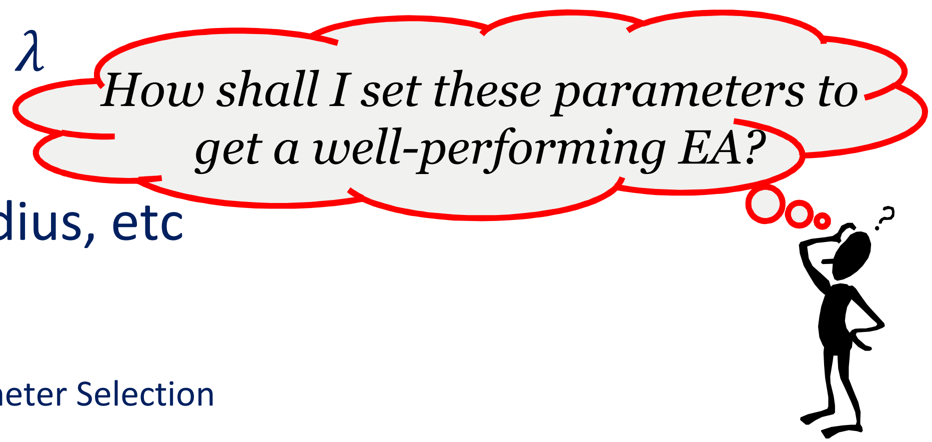
Carola Doerr, Markus Wagner: Simple On-the-Fly Parameter Selection Mechanisms for Two Classical Discrete Black-Box Optimization Benchmark Problems

The Parameter Selection Problem

- Evolutionary algorithms and related iterative optimization heuristics are parametrized algorithms
  - Example: $(\mu + \lambda)$ EAs

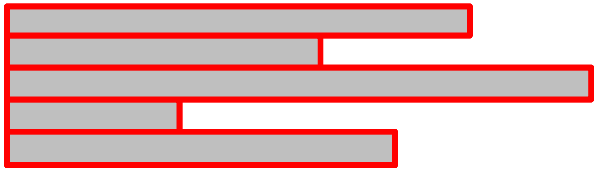


  - Parameters:
    - Memory size μ
    - Offspring population size λ
    - Crossover rate
    - Mutation rate, search radius, etc
    - Selective pressure



Parameter Tuning vs. Parameter Control

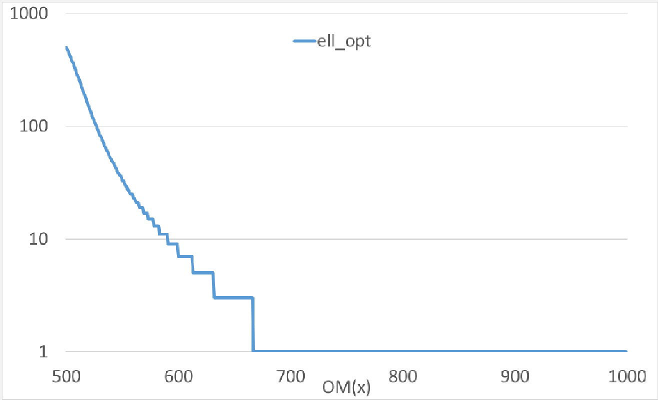
- **Parameter Tuning:**
  - Initial set of experiments
  - Deduce reasonable parameter settings



➔Does not have to be done manually, but a number of powerful, ready-to-use tools available: irace, SPOT, ParamILS, SMAC, GGA,…

- **Parameter Control:**
- 2 main differences:
  - Parameters are set *while* optimizing
  - Parameters *change over time*:

    Key motivation: different parameter values can be optimal in different stages of an optimization process



Goals of Parameter Control

- ✓ to **identify** good parameter values "on the fly"
- ✓ to **track** good parameter values when they change during the optimization process

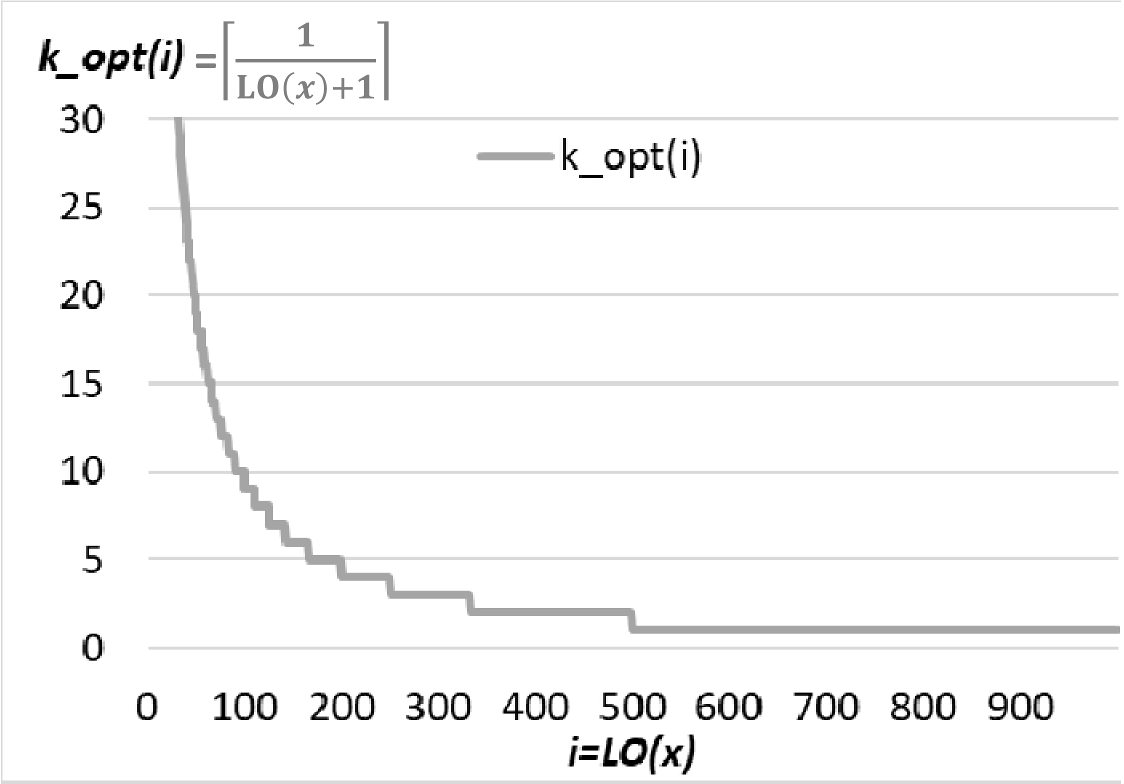
Parameter Control

- Example: LeadingOnes: LO(110110101010)=2
- Randomized Local search: flip k bits, keep the better of parent and offspring

$$k_{\mathrm{opt}}(x) = \left\lceil \frac{1}{\mathrm{LO}(x)+1} \right\rceil$$

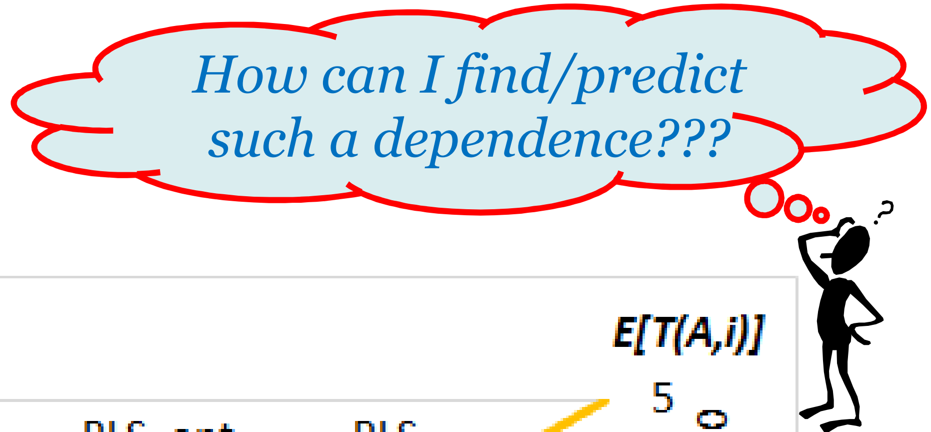
Parameter Control

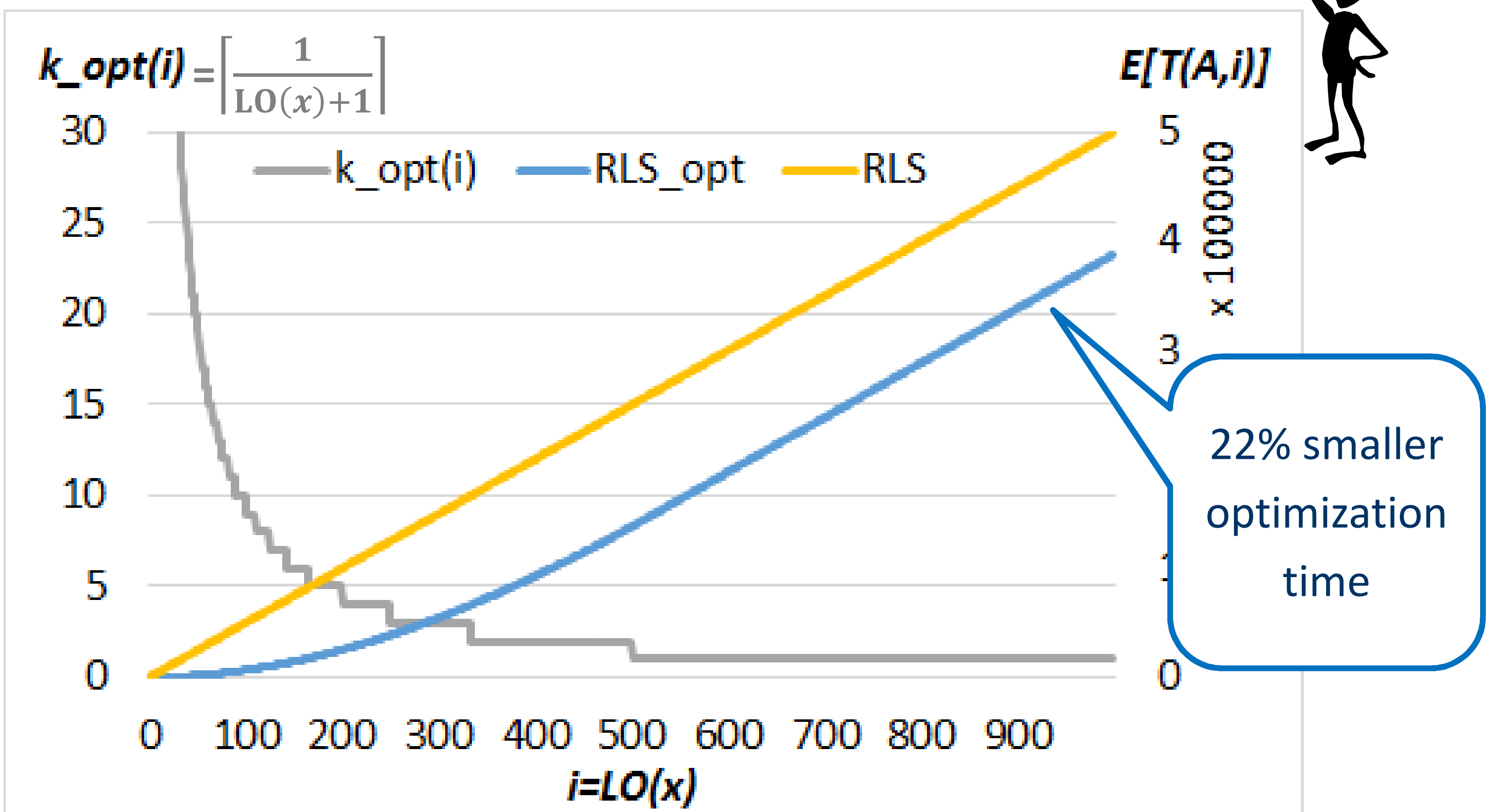
- Example: LeadingOnes: LO(110110101010)=2
- Randomized Local search: flip k bits, keep the better of parent and offspring
- n=1000



Parameter Control

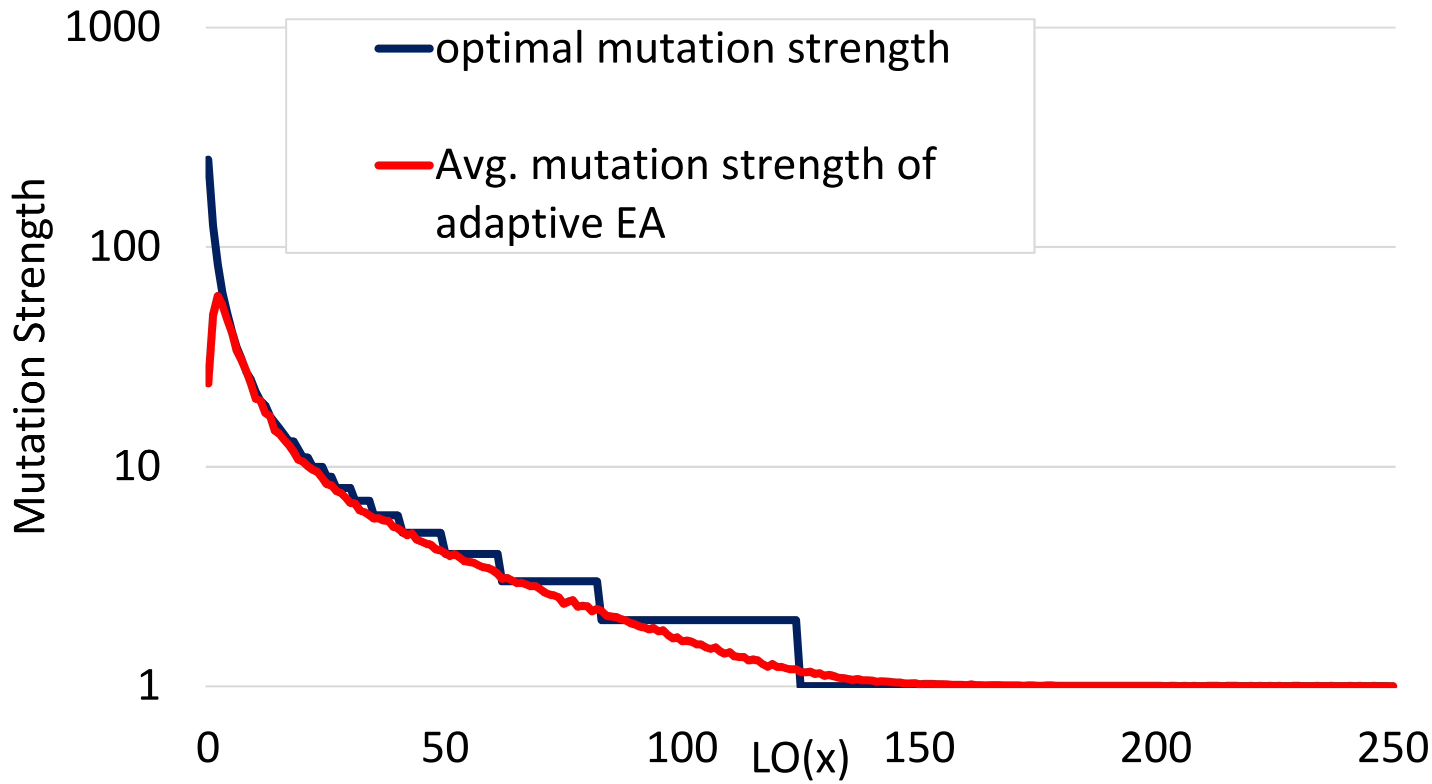
- Example: LeadingOnes: LO(110110101010)=2
- Randomized Local search: flip
  offspring
- n=1000





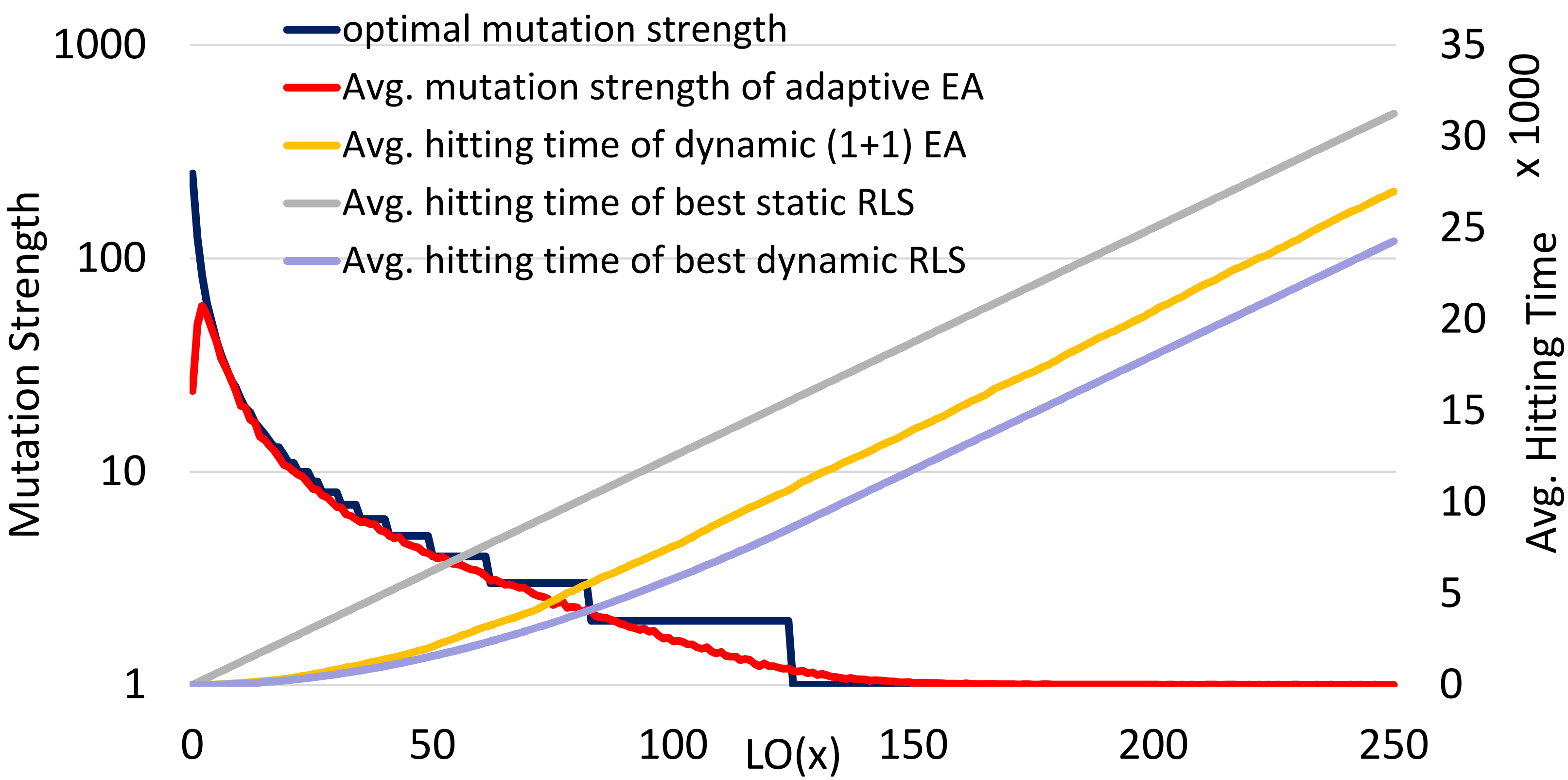
Good News: You Don't Have to!

- Easy mechanisms which find close-to-optimal parameter values automatically:



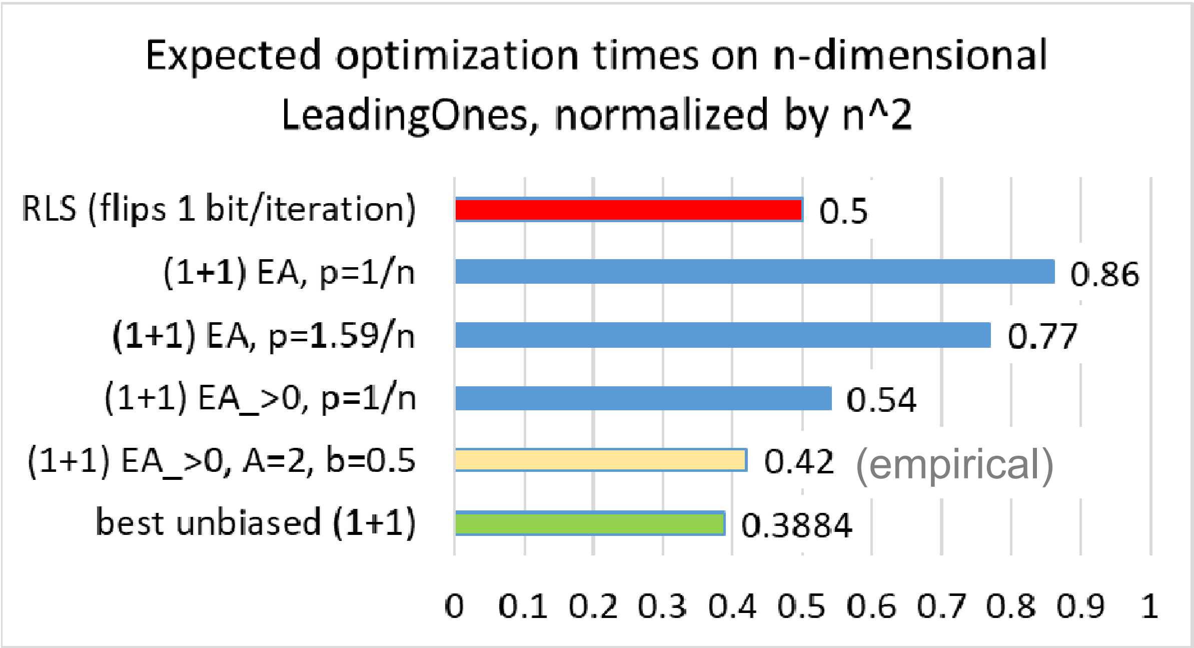
Good News: You Don't Have to!

- With close-to-optimal performance:



Good News: You Don't Have to!

- Running time for update strengths $A = 2$, $b = 1/2$



  - around 20.5% performance gain over the (1+1) $\mathrm{EA}_{>0}$ with static mutation rate $p = 1/n$
  - 14% performance gain over RLS
- larger gains possible for other combinations of A and b

Success-Based Multiplicative Update Rule

Algorithm 2: The $(1+1)$ EA_α with update strengths A and b and initial mutation rate $p_0 \in [1/n^2, 1/2]$ 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 $f(x)$;

Set $p = p_0$;

Optimization: for $t = 1, 2, 3, \ldots$ **do**

- Create offspring y through standard bit mutation with mutation probability p
- **if** $f(y) \geq f(x)$ **then**
  - $x \leftarrow y$ and $p \leftarrow \min\{A \cdot p, 1/2\}$
- **else**
  - $p \leftarrow \max\{b \cdot p, 1/n^2\}$

A>1

b<1

Success-Based Multiplicative Update Rule

Algorithm 2: The $(1+1)$ EA_α with update strengths A and b and initial mutation rate $p_0 \in [1/n^2, 1/2]$ for the maximization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$

```
Initialization: Sample x ∈ {0,1}^n uniformly at random
    and compute f(x);
Set p = p_0;
Optimization: for t = 1, 2, 3, ... do
    Sample ℓ from Bin_{>0}(n, p);
    y ← flip_ℓ(x);
    evaluate f(y);
    if f(y) ≥ f(x) then
        x ← y and p ← min{A · p, 1/2}
    else
        p ← max{b · p, 1/n^2}
```

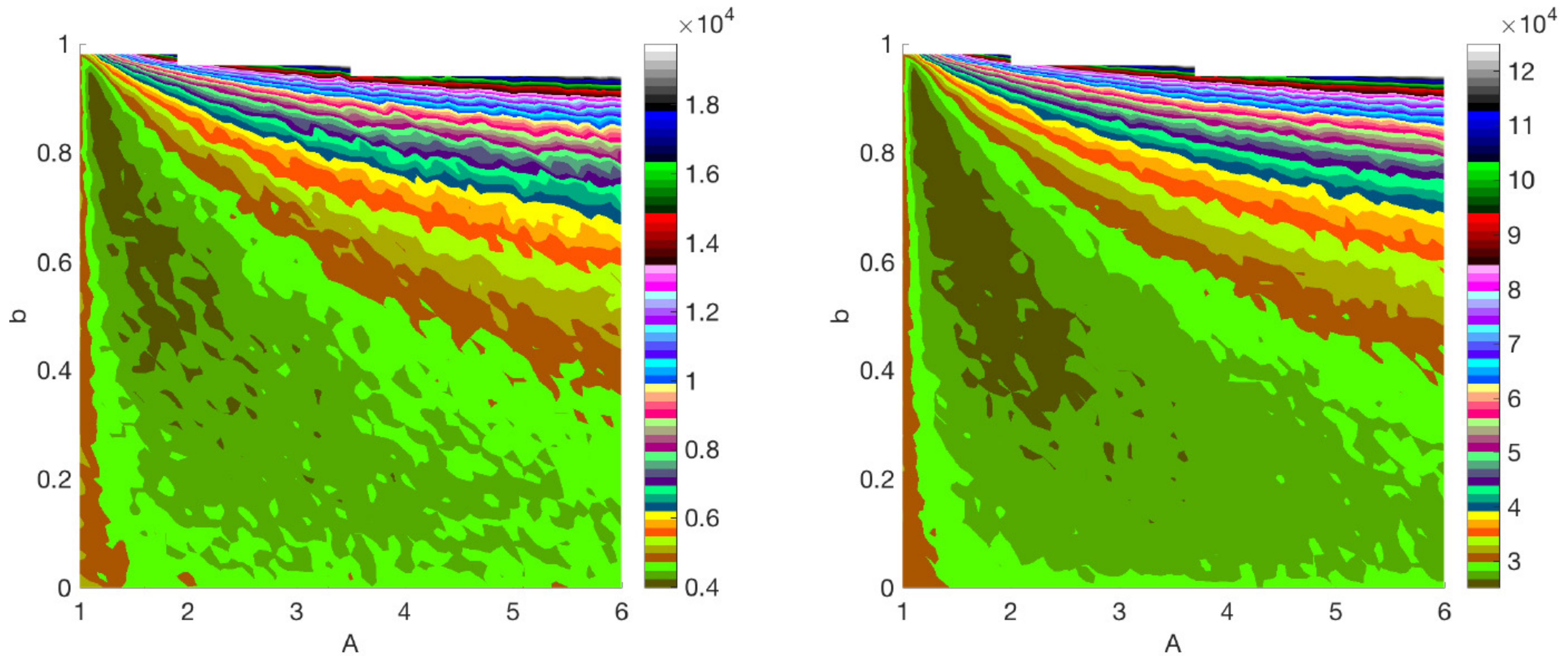
Standard bit mutation, condition to flip at least one bit

A>1

b<1

LeadingOnes

- Average optimization time for different combinations of A and b (101 independent runs)

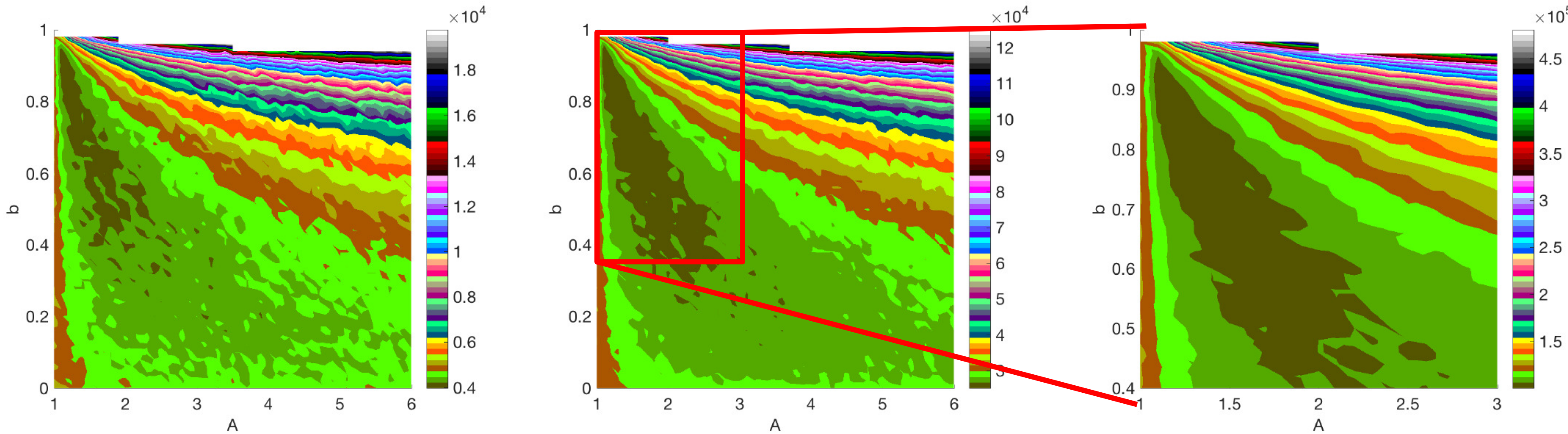


(d) **LeadingOnes with** $n = 100$ (e) **LeadingOnes with** $n = 250$

- For comparison: RLS needs $n^2/2$ iterations (=0.5 and =3.125 above), (1+1) $EA_{>0}$ needs 0.54 and 3.4 * 10^4 iterations, respectively

LeadingOnes

- Average optimization time for different combinations of A and b (101 independent runs)

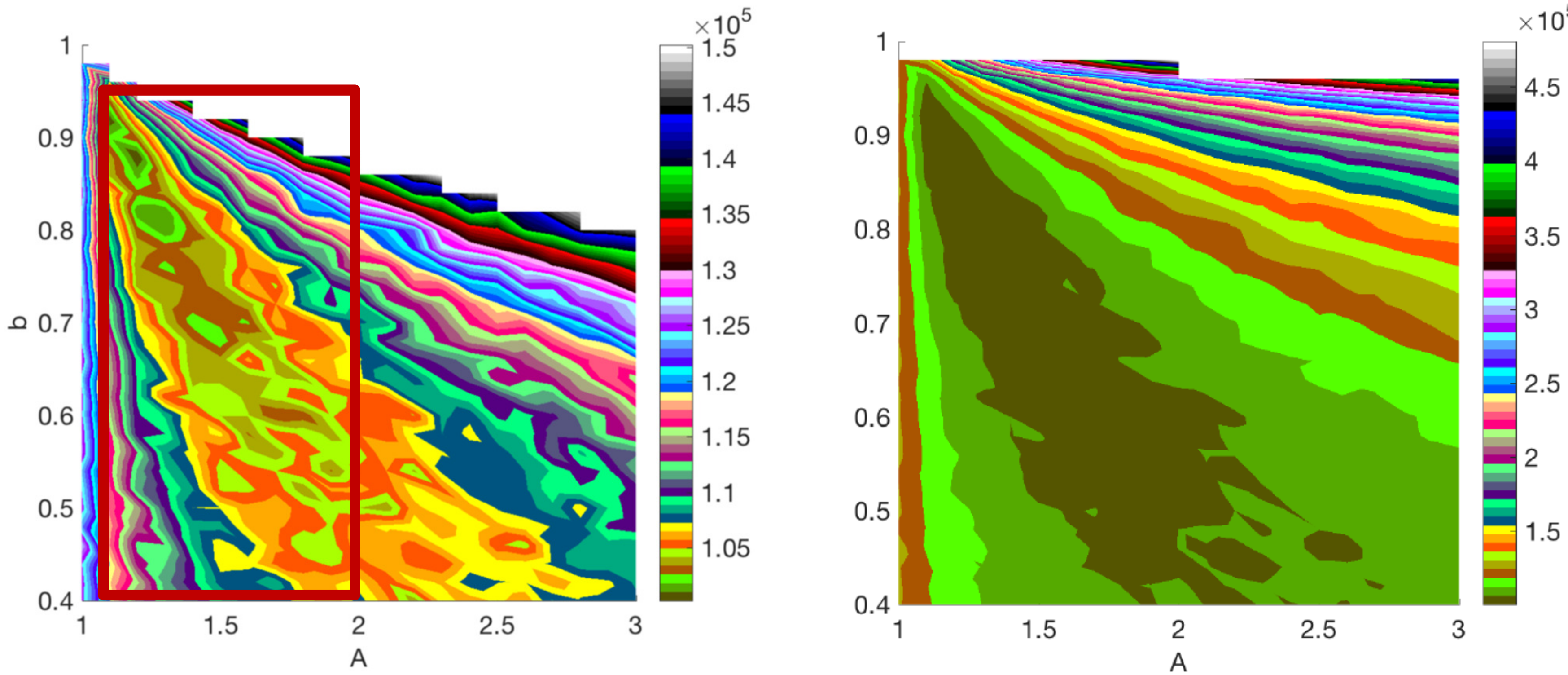


(d) LeadingOnes with $n = 100$ (e) LeadingOnes with $n = 250$ (f) LeadingOnes with $n = 500$

- For comparison: RLS needs $n^2/2$ iterations (=0.5, =3.125, 1.25 above), (1+1) $EA_{>0}$ needs 0.54, 3.4 * 10^4, and 1.35*10^5 iterations, respectively

LeadingOnes

- Average optimization time for different combinations of A and b (101 independent runs)



(f) LeadingOnes with $n = 500$

- For comparison: RLS needs $n^2/2$ iterations (=1.25*10^5 for n=500), (1+1) $EA_{>0}$ needs 1.35*10^5 iterations, respectively

1/5-th Success Rules

- 1/5-th success rule:
  - originally from continuous optimization [Rechenberg, Devroye, Schumer/Steiglitz]
  - (1+1) ES optimizing sphere $f(x) = \sum x_i^2$
  - When success rate > 1/5: increase search radius
    When success rate < 1/5: decrease search radius
- In discrete optimization, e.g., [Kern/Müller/Hansen/Büche/Ocenasek/Koumoutsakos04, Auger09]:
  - When success rate $\approx 1/5$, parameter value should be stable
  - In our algorithm:

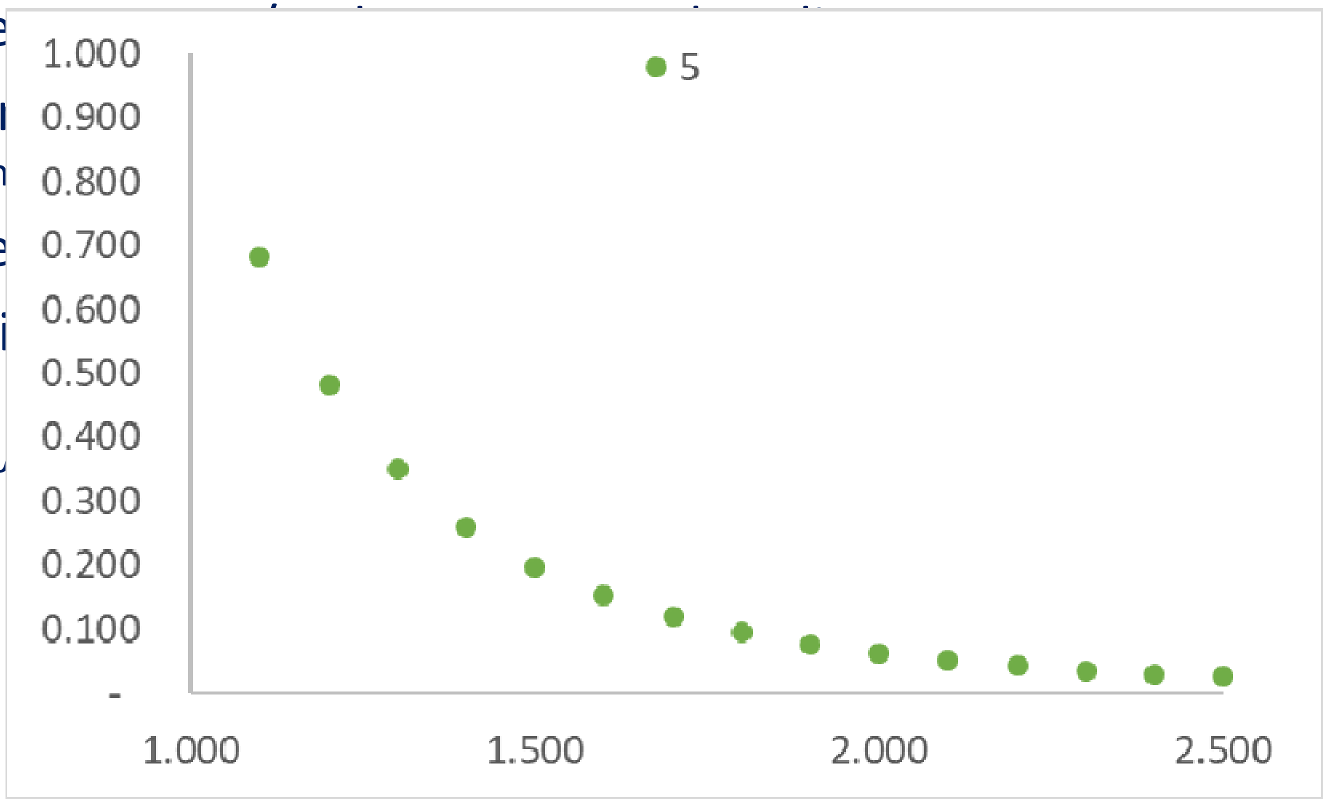
    If $f(y) \geq f(x)$: $p \leftarrow \min\left\{Ap, \frac{1}{2}\right\}$
    else $p \leftarrow \max\{bp, 1/n^2\}$
  - $A = \left(\frac{1}{b}\right)^{1/4}$ since $Ab^4 = 1$
    $b = 1/A^4$

1/5-th Success Rules

- 1/5-th success rule:
  - originally from continuous optimization [Rechenberg, Devroye, Schumer/Steiglitz]
  - (1+1) ES optimizing sphere $f(x) = \sum x_i^2$
  - When success rate > 1/5: increase search radius
    When succe
- In discrete opti
  [Kern/Müller/Hansen
  - When succe
  - In our algori

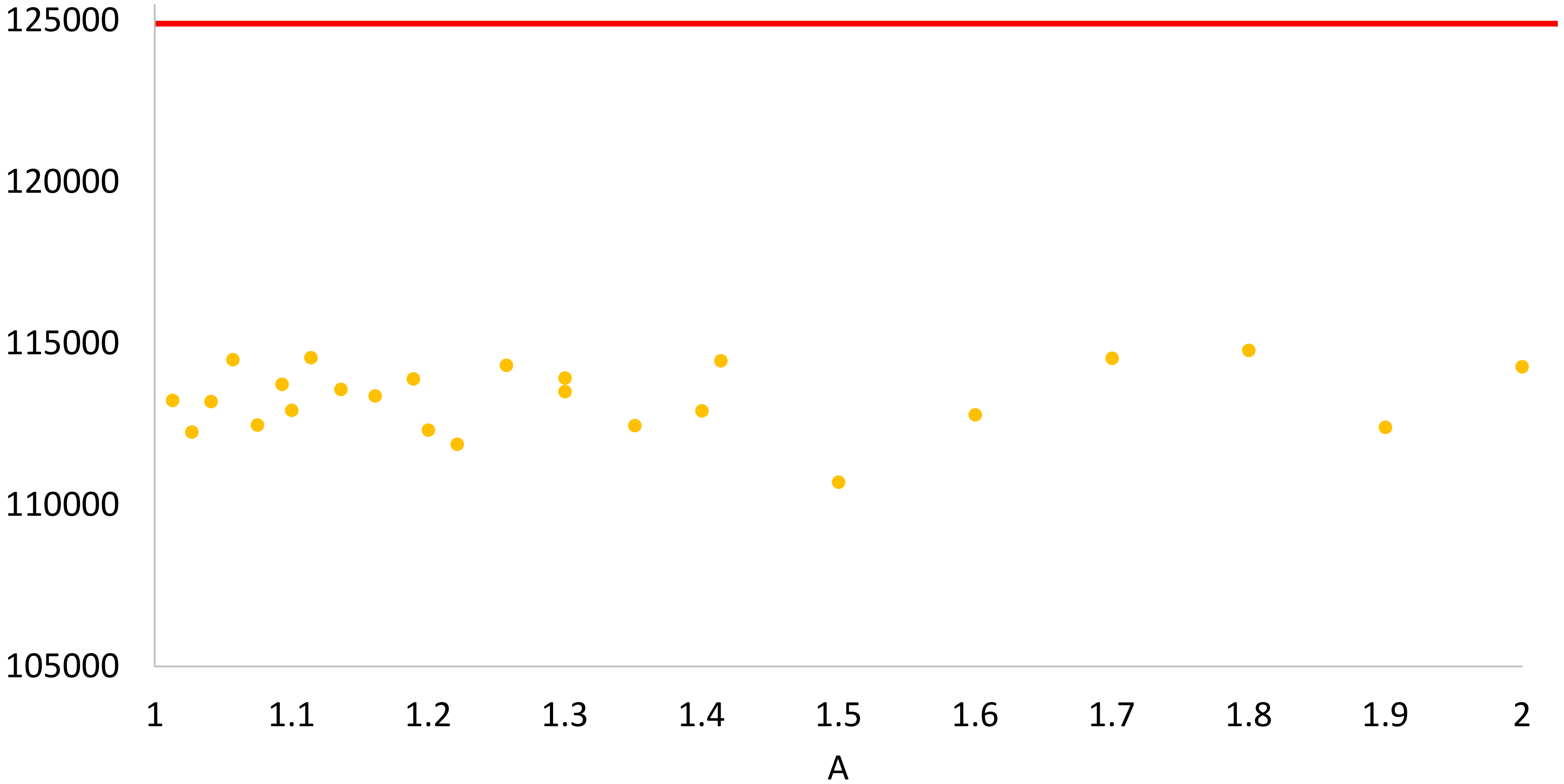
    If $f(y) \geq$

    else
  - $A = \left(\frac{1}{b}\right)^{1/4}$
    $b = 1/A^4$



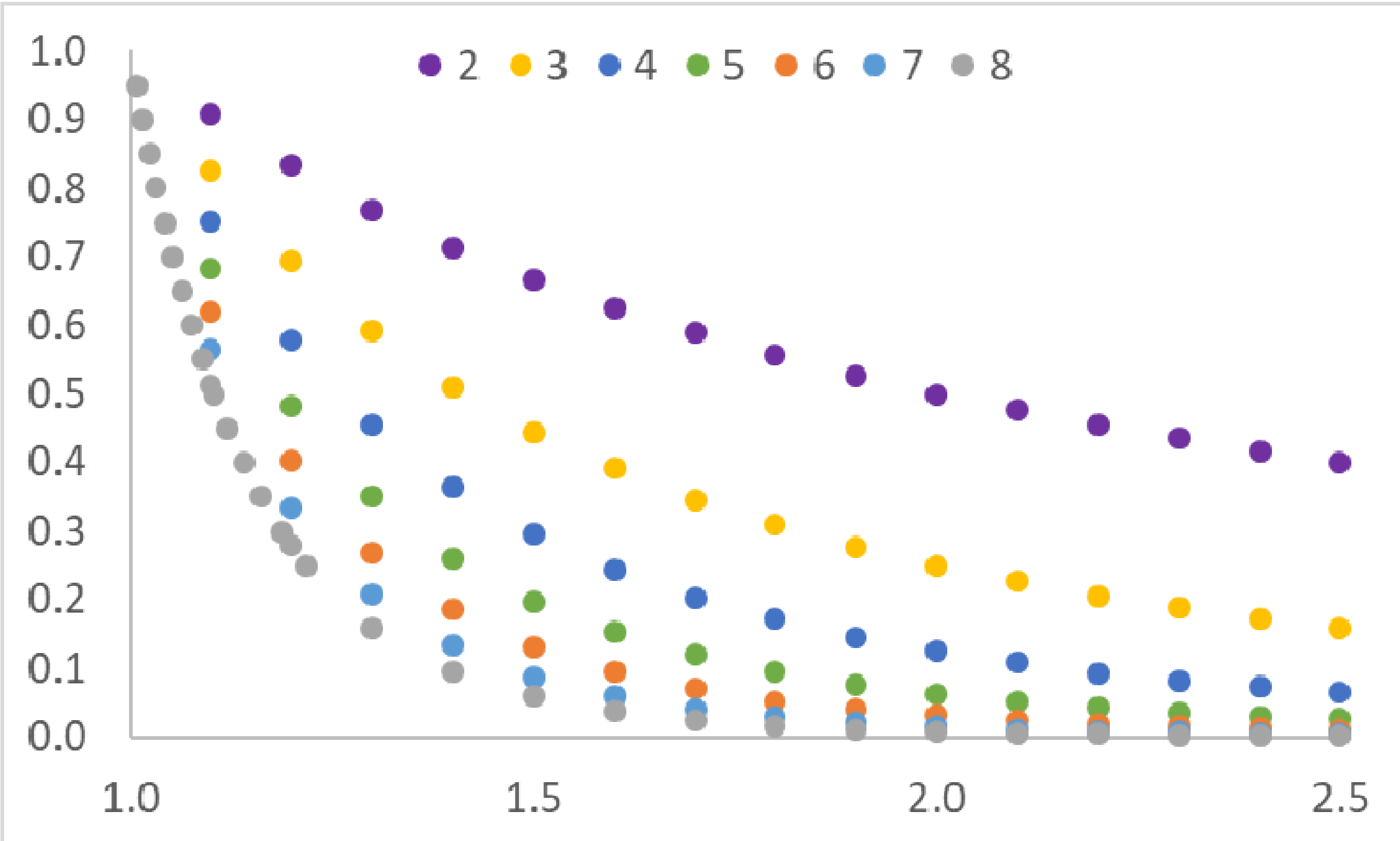
Results for the 1/5-th Success Rule

- LO, n=500, 100 independent runs
- RLS performance: 125,000 iterations



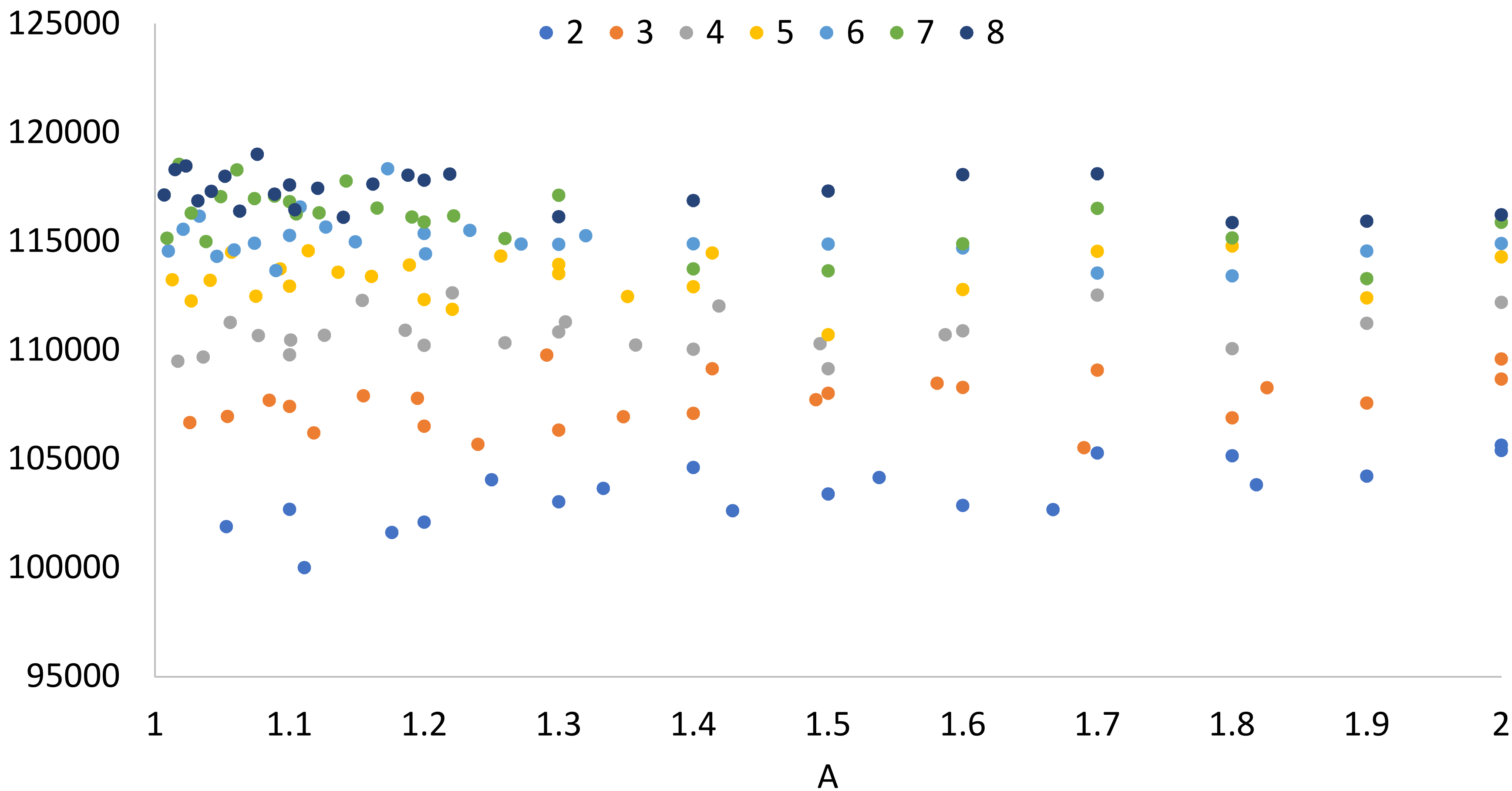
1:x Success Rules

- A priori no reason the restrict ourselves to a 1:5 success ratio
- We can also try different success rules



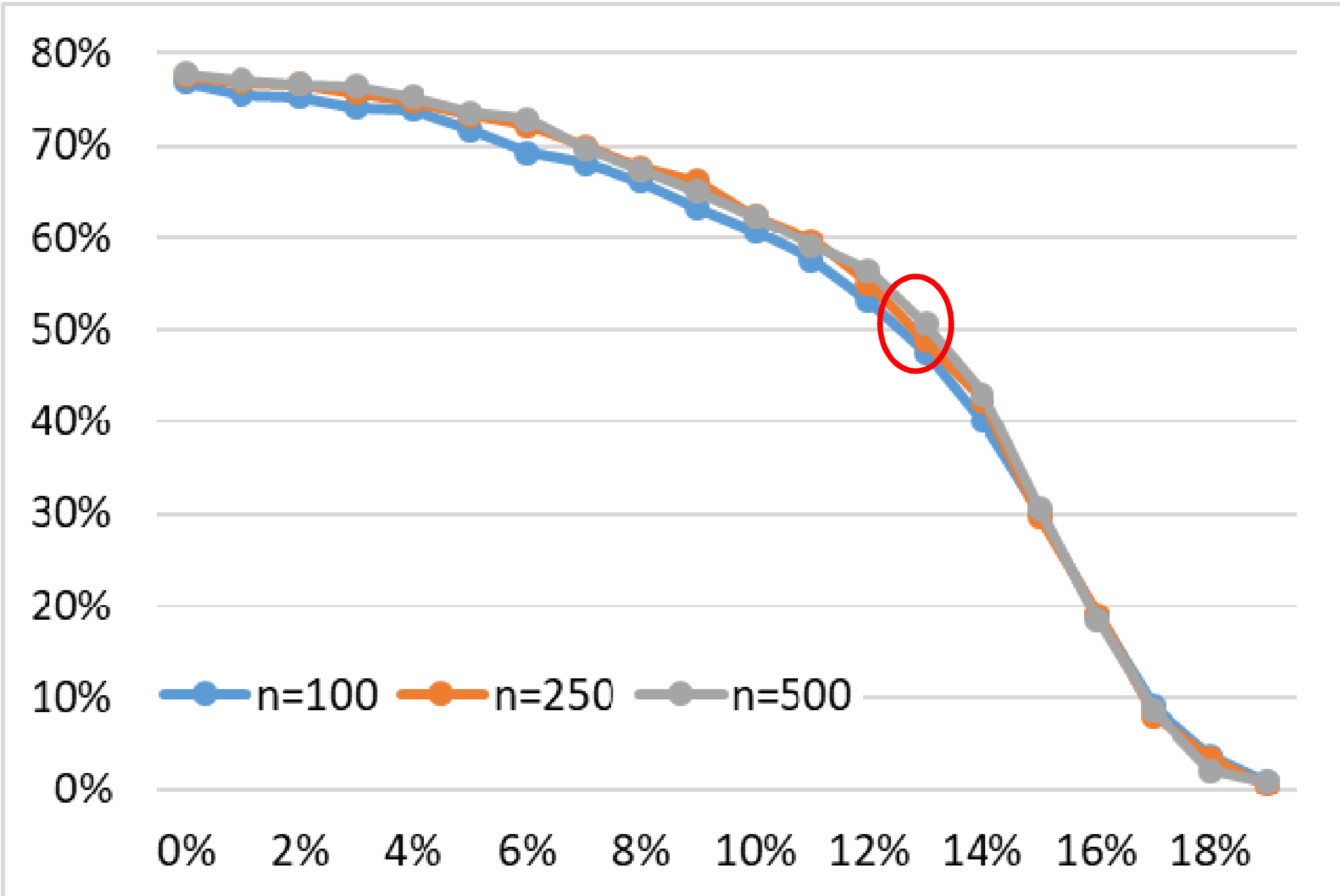
Average Optimization Times of 1:x Rules

- LO, n=500, 100 independent runs
- RLS performance: 125,000 iterations

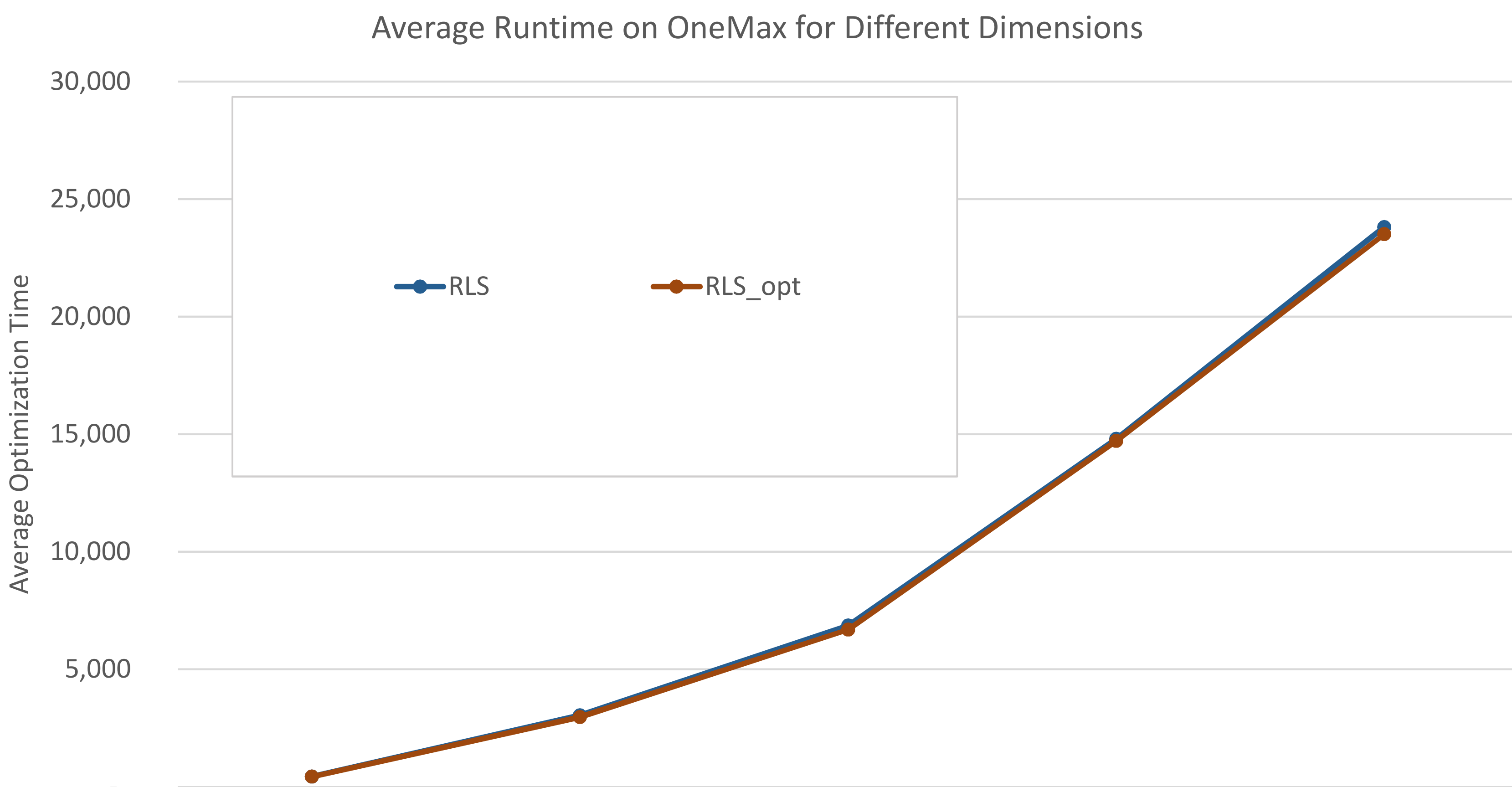


Overall Performance Summary

- 50% of all configurations with $1 < A \leq 2.5$ and $0.4 \leq b < 1$ are better than RLS by at least 13%



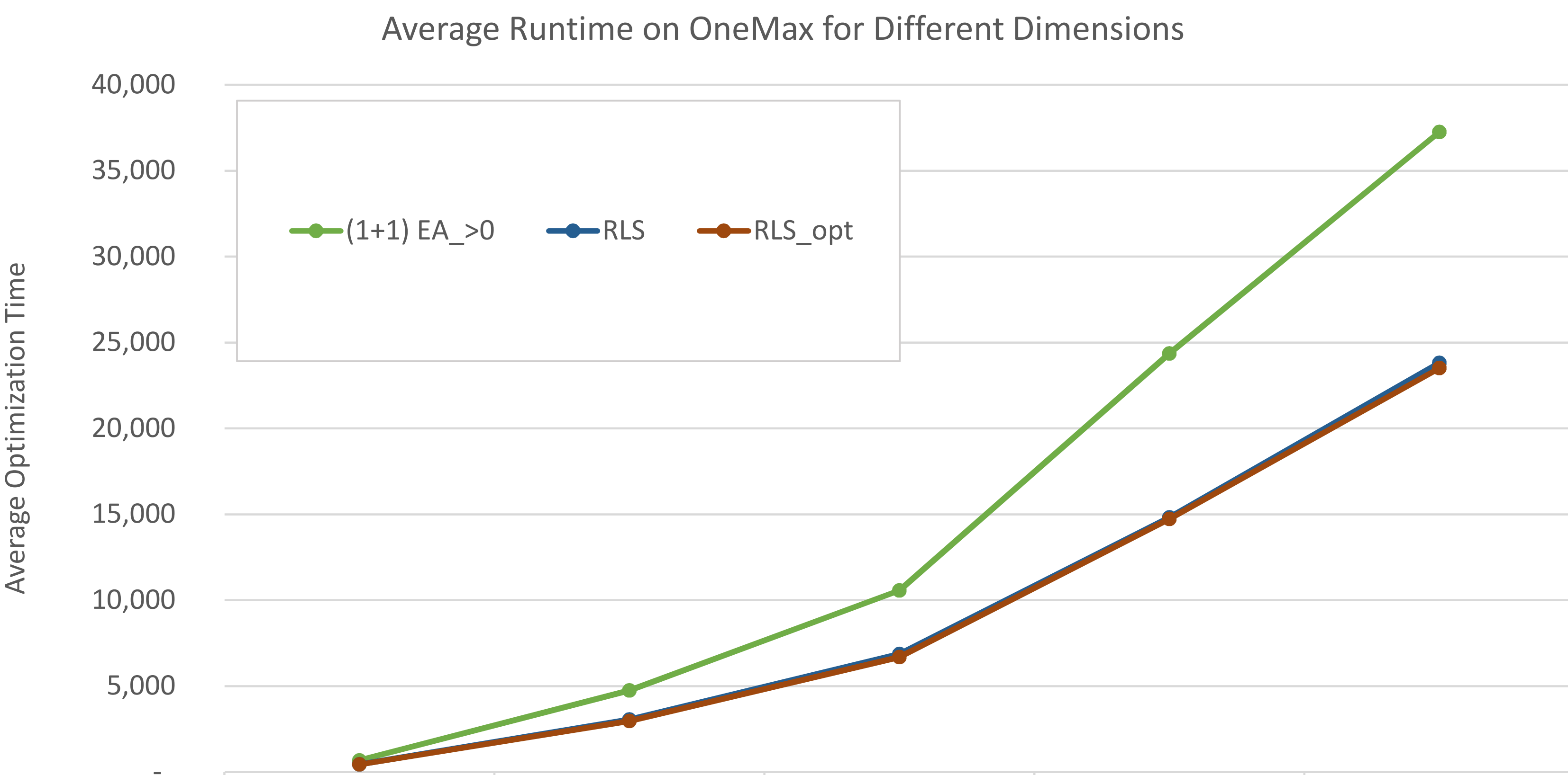
Results for OneMax



	100	500	1000	2000	3000
RLS	445	3,050	6,871	14,809	23,814
RLS_opt	436	2,974	6,690	14,722	23,507

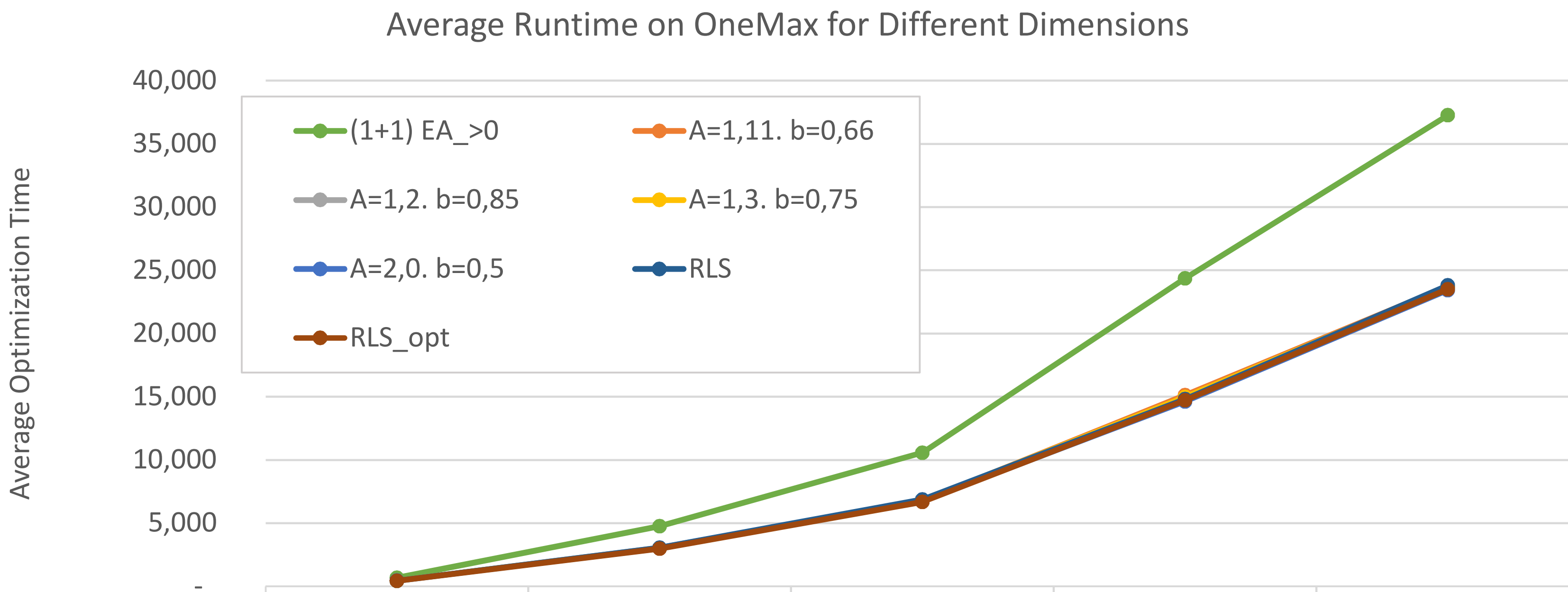
Dimension n

Results for OneMax



	100	500	1000	2000	3000
(1+1) EA_>0	679	4,756	10,574	24,352	37,256
RLS	445	3,050	6,871	14,809	23,814
RLS_opt	436	2,974	6,690	14,722	23,507

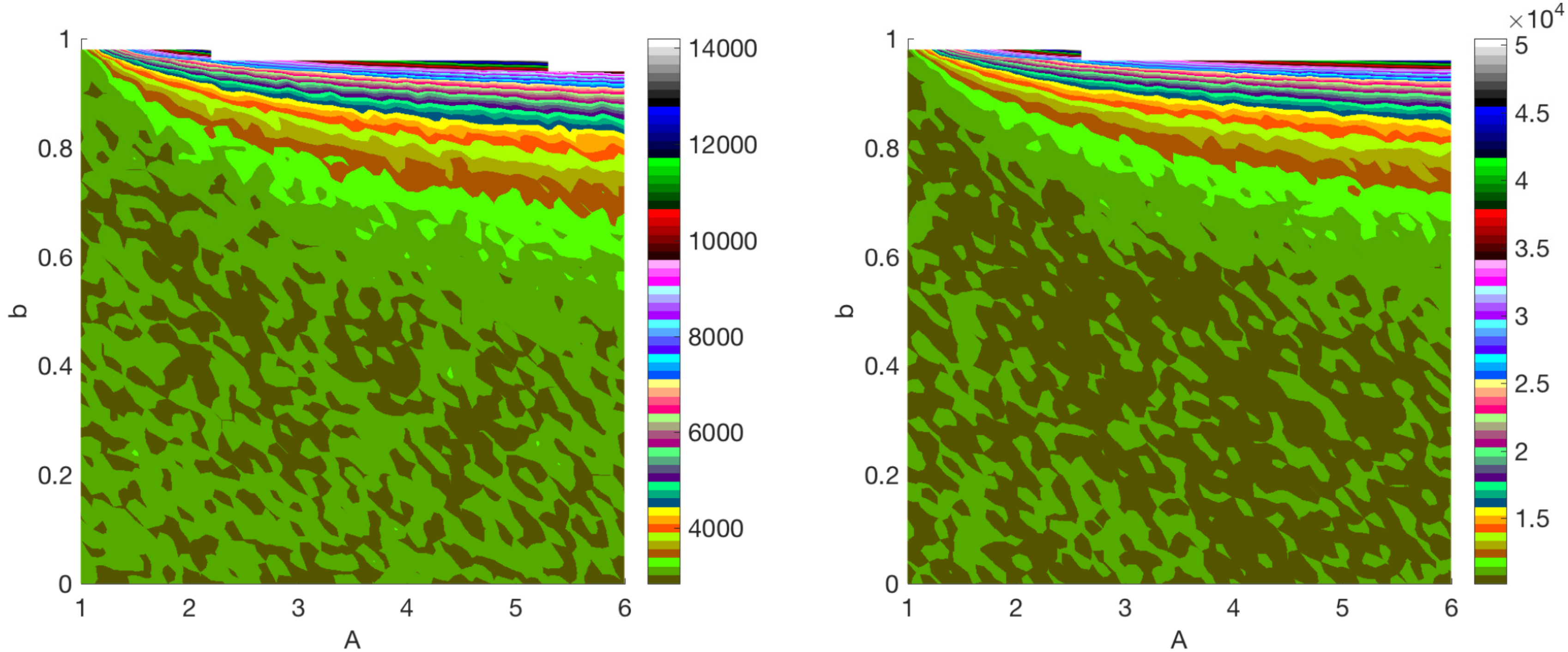
Results for OneMax



	100	500	1000	2000	3000
(1+1) EA_>0	679	4,756	10,574	24,352	37,256
A=1,11. b=0,66	447	3,039	6,749	15,134	23,726
A=1,2. b=0,85	450	3,059	6,751	14,801	23,558
A=1,3. b=0,75	450	3,033	6,801	14,974	23,715
A=2,0. b=0,5	455	3,013	6,753	14,613	23,417
RLS	445	3,050	6,871	14,809	23,814
RLS_opt	436	2,974	6,690	14,722	23,507

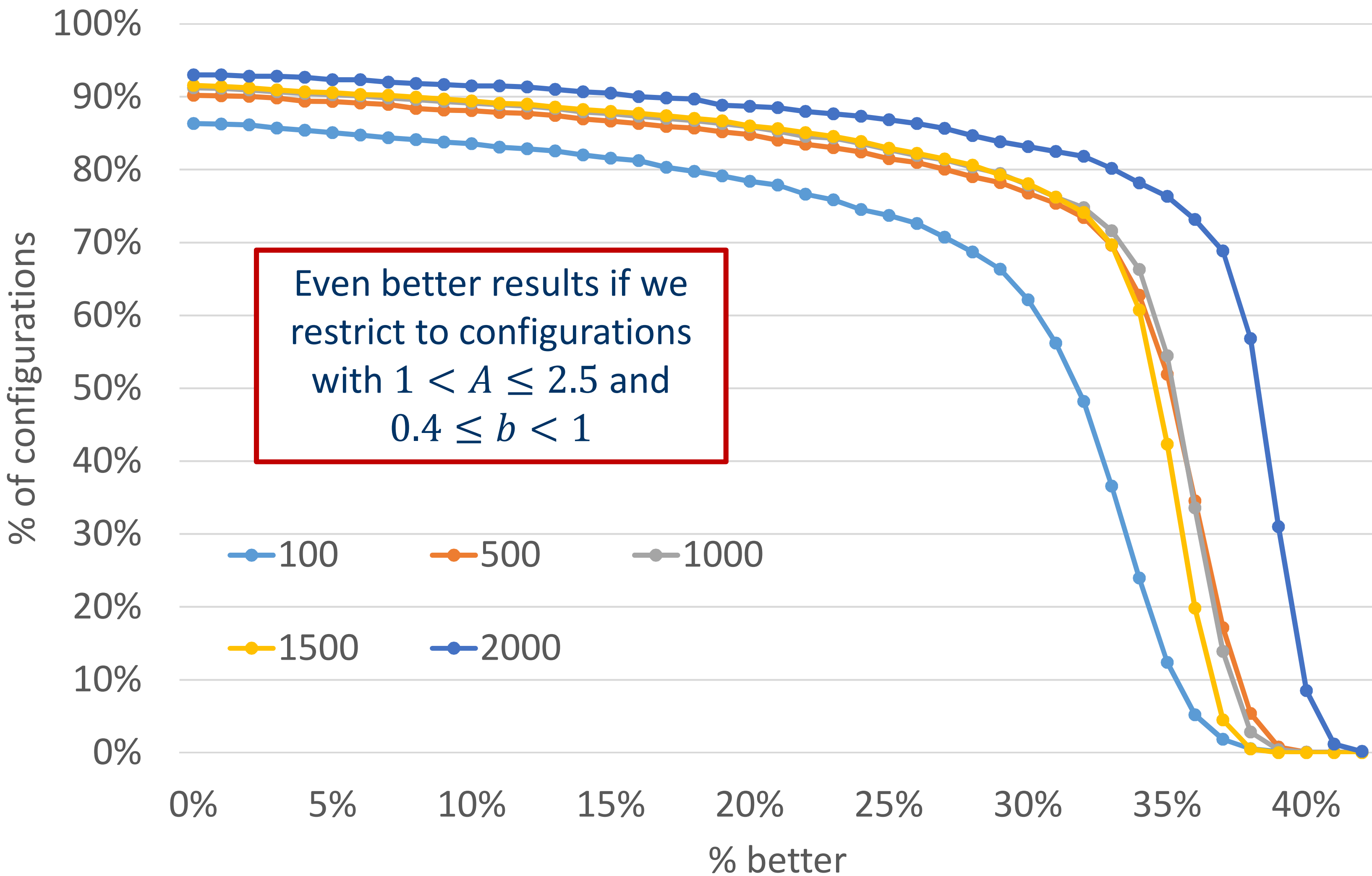
Dimension n

Heatmaps for OneMax

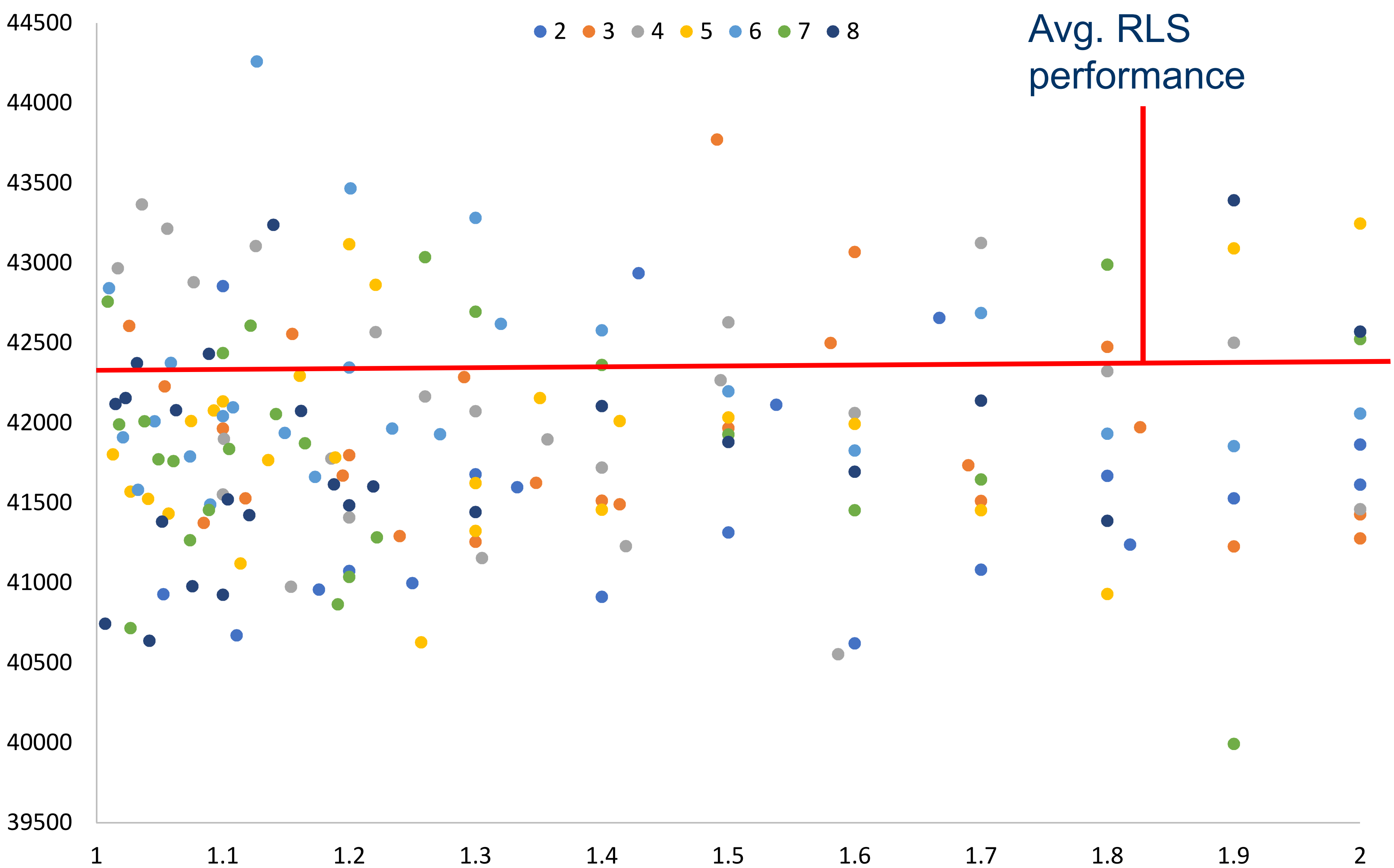


(a) OneMax with $n = 500$ **(b) OneMax with** $n = 1500$

$y\%$ Configs better than $(\mathbf{1}+\mathbf{1})\mathrm{EA}_{>\mathbf{0}}$ by at least $x\%$



1:x Rules, OneMax, n=5000, 100 independent runs



Next Steps

- Theoretical performance guarantees for the adaptive (1+1) EA_{α}
- Comparison with other adaptation schemes, e.g.,
  - Adaptive Pursuit [Thierens 05]
  - UCB algorithms from Machine Learning [Da Costa, Fialho, Schoenauer, Sebag 08-11]
  - ε-greedy algorithm from [Doerr, Doerr, Yang 16]
- Performance on other test functions
  - Real-world problems?
    → you are all cordially invited to collaborate on this!
- Want to know more about dynamic parameter choices?
  → confer the tutorial slides (available on my homepage)

Acknowledgments

- We thank Eduardo Carvalho Pinto for providing his implementation of the (1+1) EA$_{\alpha}$ and his contributions to a preliminary experimentation with the multiplicative parameter control mechanism.
- Our work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, and by the Australian Research Council project DE160100850.