

Step size selection in Frank Wolf

Overview

Problem statement

Minimize $f(\mathbf{x})$

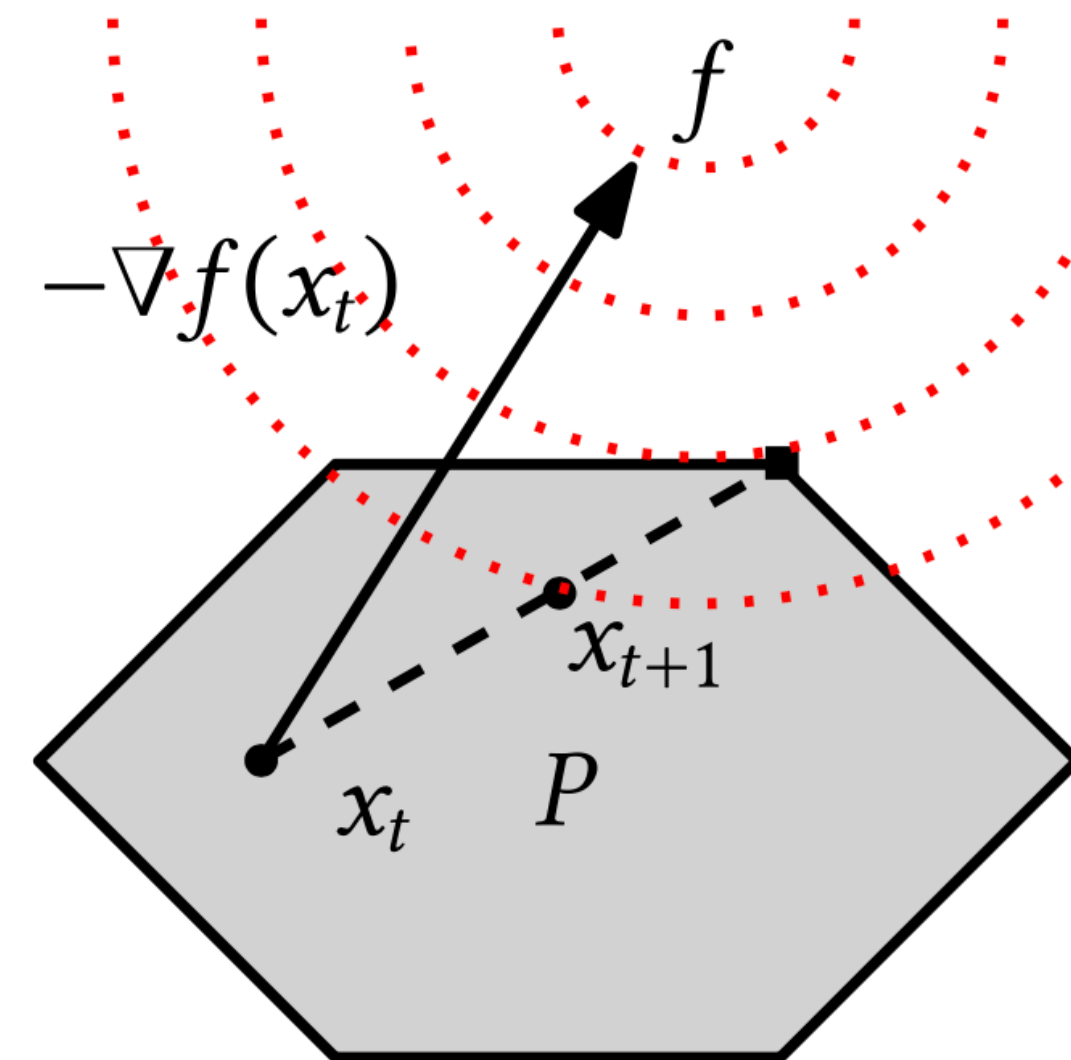
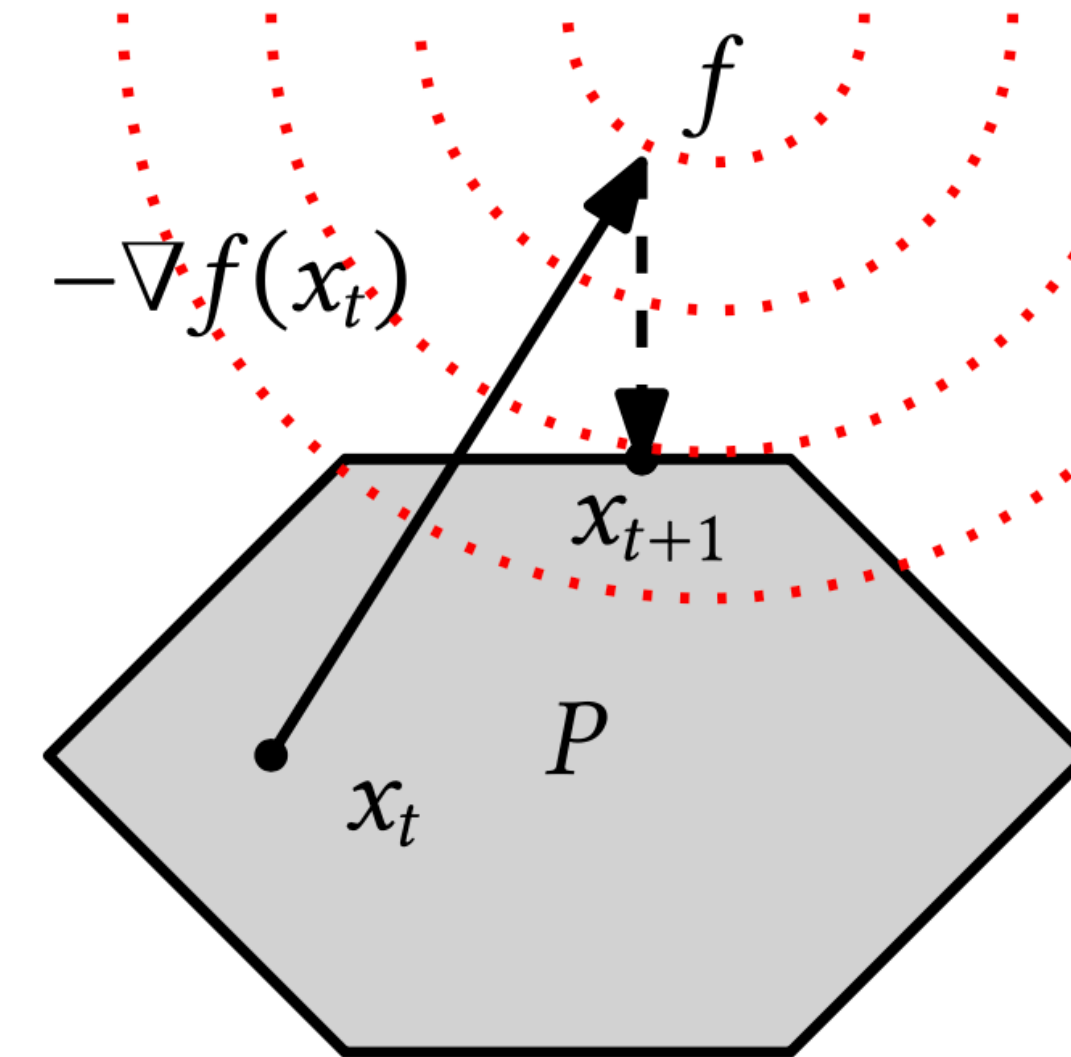
Subject to $\mathbf{x} \in \mathcal{D}$

- \mathcal{D} - **compact, convex** set in a vector space
- $f: \mathcal{D} \rightarrow \mathbf{R}$ is a **convex, L -smooth**, function

Linear Minimization Oracle

$$s = \arg \min_{s \in \mathcal{D}} s^T \nabla f(x_k)$$

- Finds a vector s in feasible set \mathcal{D} , which aligns most with $\nabla f(x_k)$.
- Vector s has the largest projection on $-\nabla f(x_k)$. Usually a vertex of the domain.



Algorithm

1. $s_k = \arg \min_{s \in \mathcal{D}} s^T \nabla f(x_k)$
 2. $x_{k+1} = (1 - \gamma_k)x_k + \gamma_k s_k$
- , where $\gamma_k \in [0,1]$ is a step-size.
- Both $x_k, s_k \in \mathcal{D}$. Convex combination of them is going to remain in the set. $x_{k+1} \in \mathcal{D}$

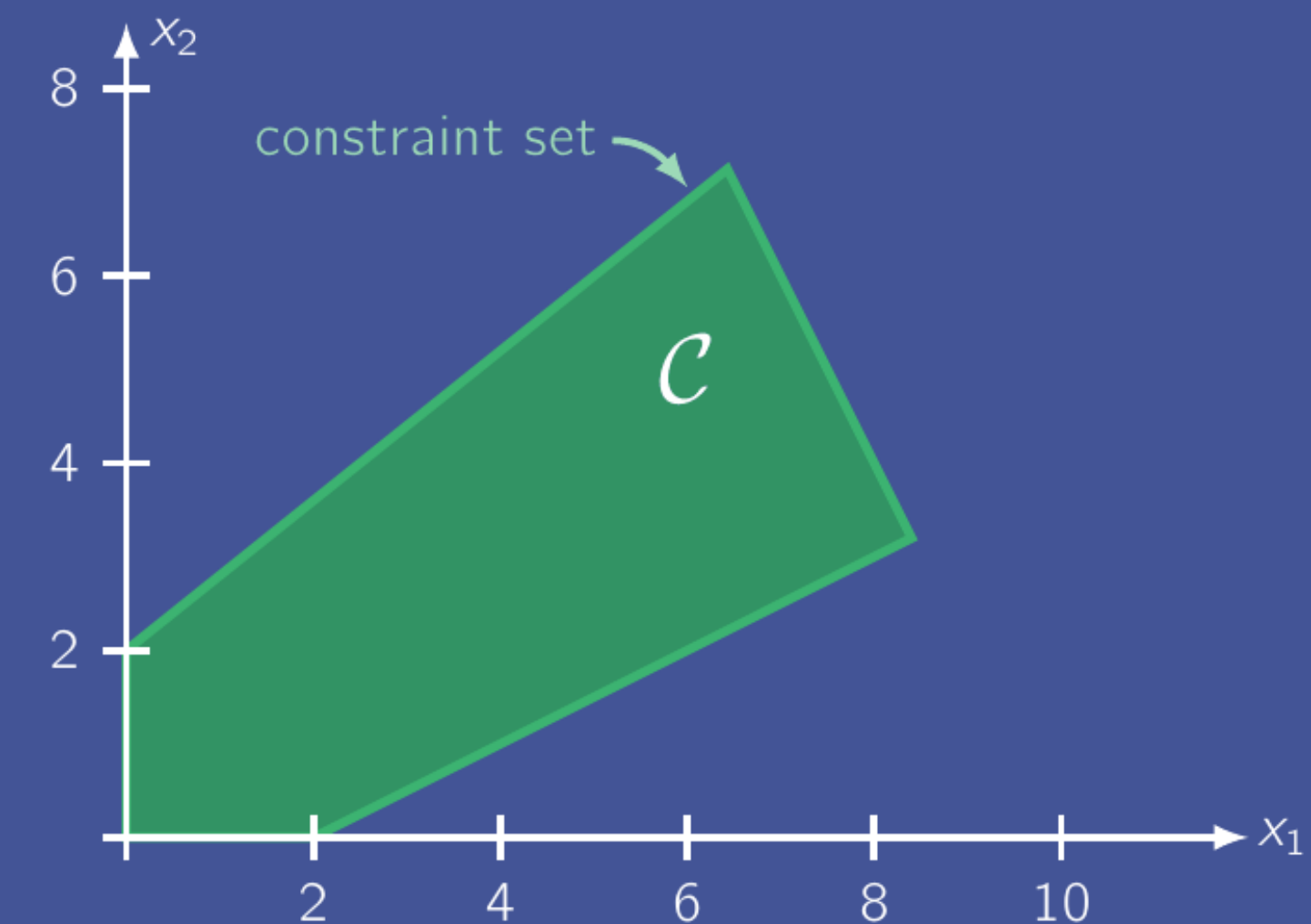
Conditional Gradient

To minimize f over \mathcal{C} , create $\{x^k\}$ via updates

$$s^{k+1} = \operatorname{argmin}_{s \in \mathcal{C}} s^T \nabla f(x^k)$$

$$x^{k+1} = x^k + \alpha_k (s^{k+1} - x^k)$$

Step size is often set to be $\alpha_k = 2/(k+2)$



Properties

- Convergence rate $O(1/k)$
- No need to do projection step (linear optimization vs. quadratic)
- Solve high dimensional problems
- Sparse solutions
- Designed for smooth, convex f
- Poor performance near optimum
- Complex, non-linear boundaries increase computation costs.

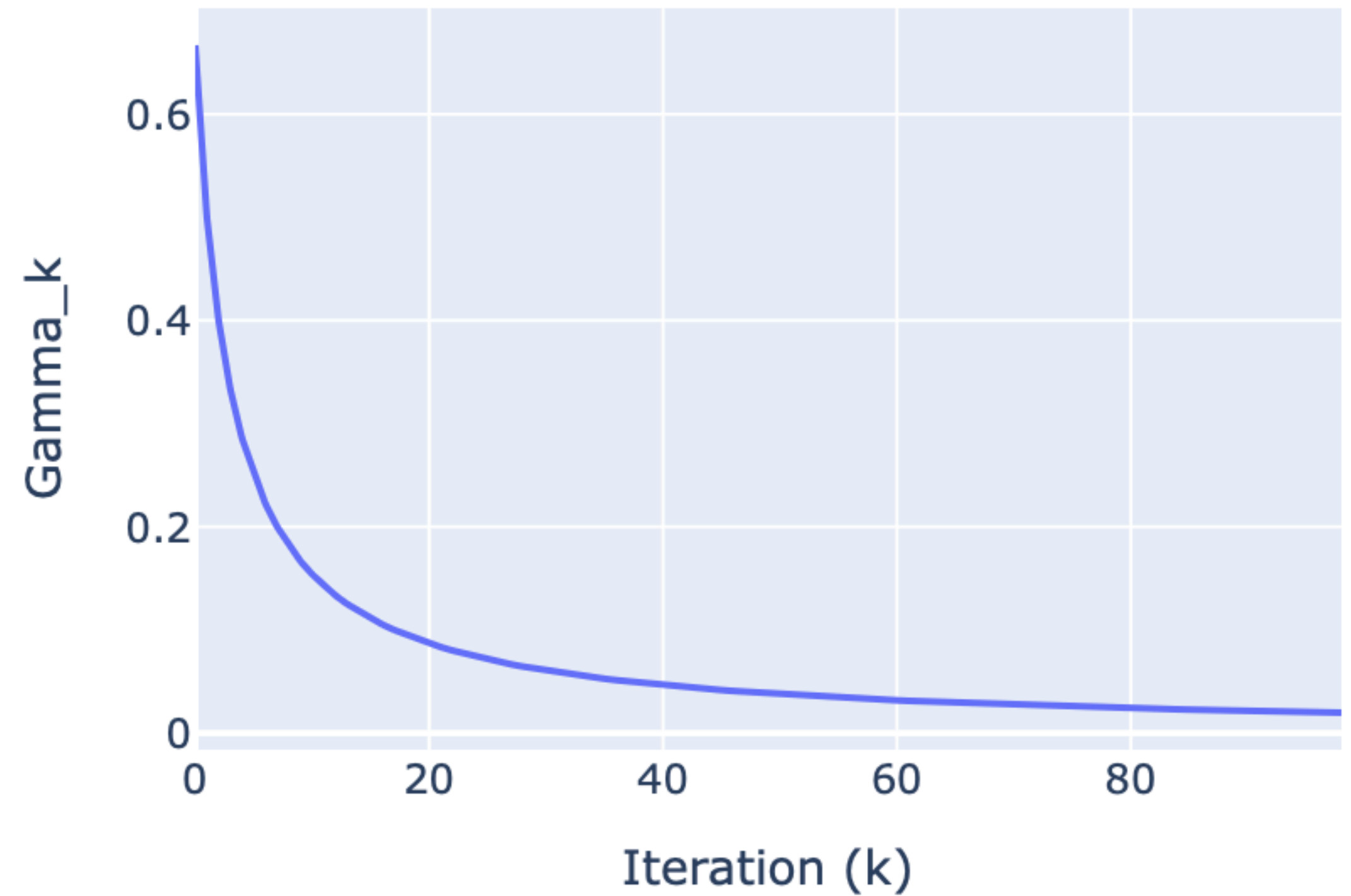
Step size

Line step size

$$\gamma_k = \frac{2}{k+2}$$

- Straightforward and cheap to compute.
- Slow convergence near optimum
- No function adaptation

Line step size

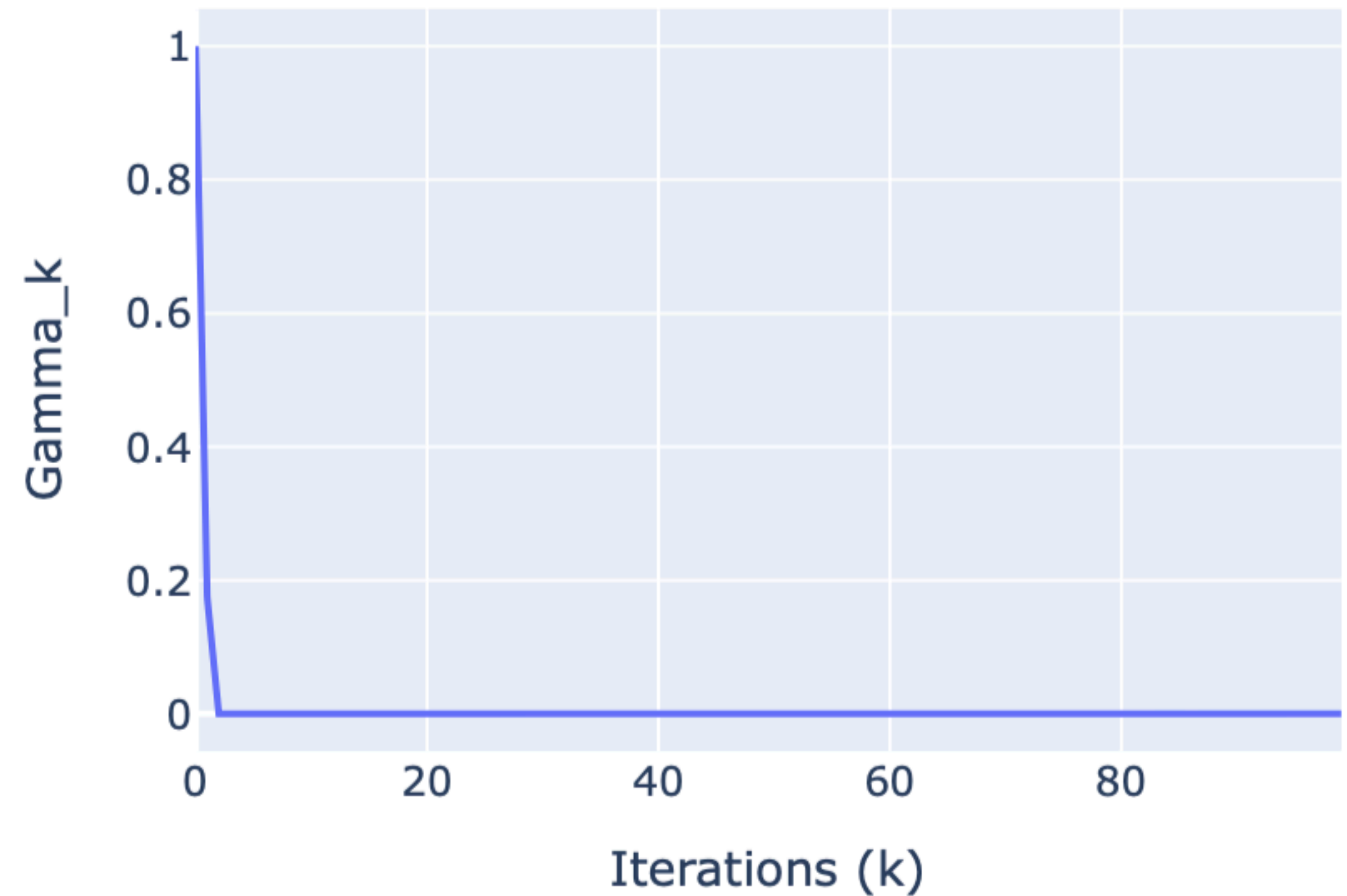


Exact line-search

$$\gamma_k = \arg \min f(x_k + \gamma_k(s_k - x_k))$$

- Ensures highest decrease per iteration
- Costly optimization problem

Exact line search

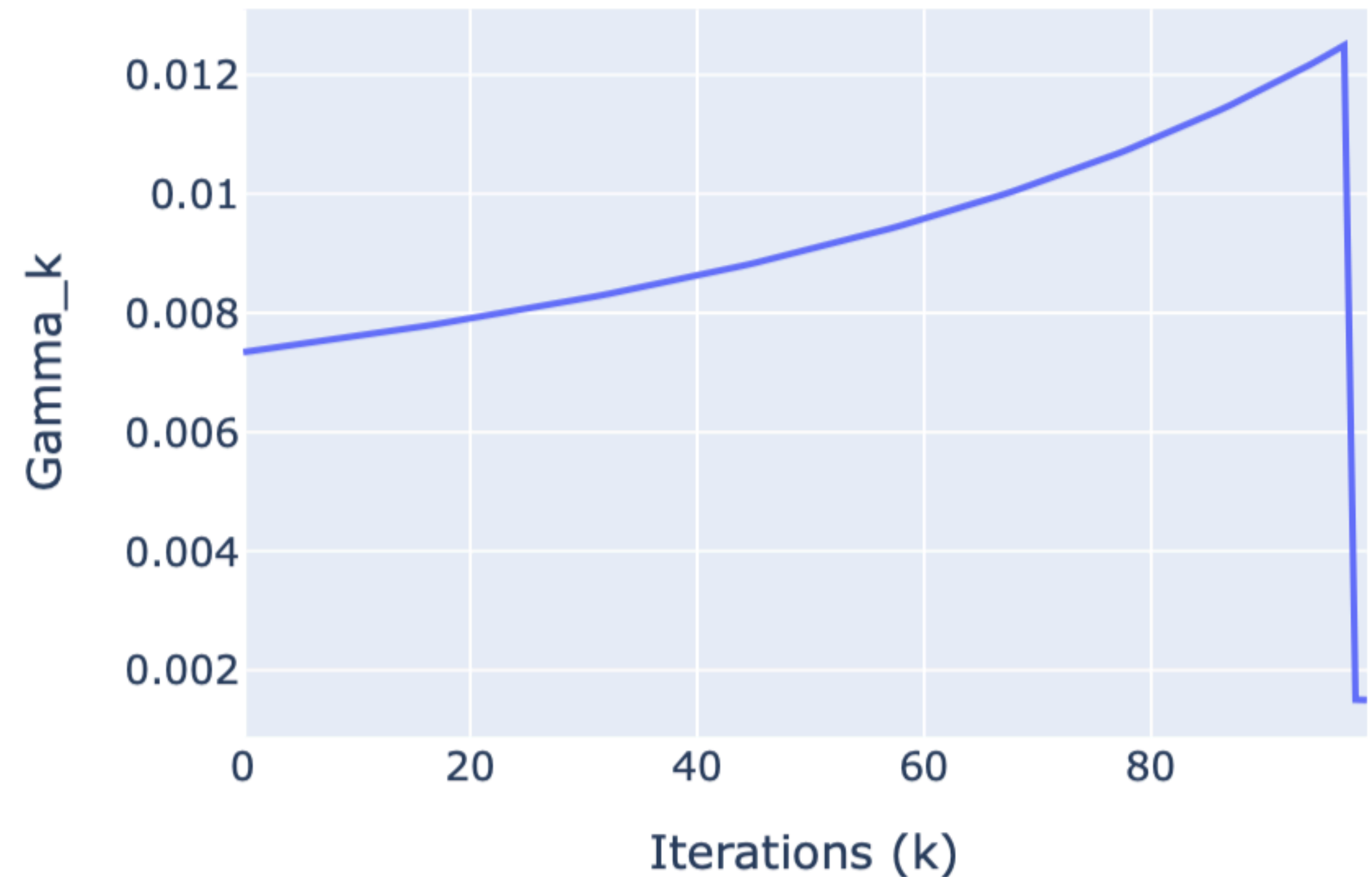


Demyanov-Rubinov

$$\gamma_k = \min\left\{\frac{-\nabla f(x)^T(s_k - x_k)}{L\|s_k - x_k\|^2}, 1\right\}$$

- Goes to zero as we approach the optimum
- Responsive for geometry of f
- Require access to L
- Unstable for small denominator (near optimum)

Demyanov Rubinov



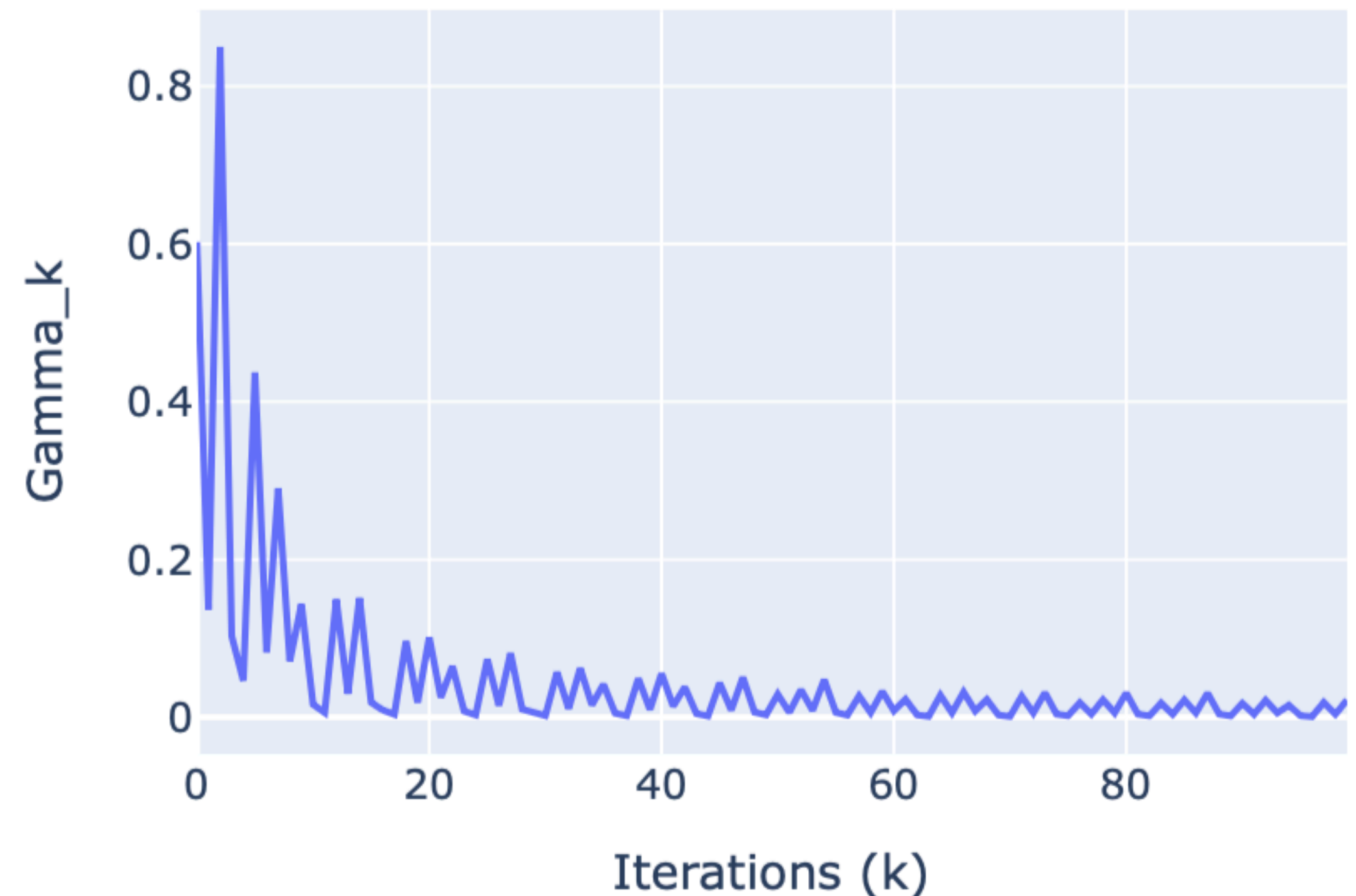
Backtracking line search

$$\gamma_k = \min\left\{\frac{-\nabla f(x)^T(s_k - x_k)}{M\|s_k - x_k\|^2}, 1\right\}$$

M_t is approximation of L

- Doesn't require L
- Adaptive and stable progress
- Require multiple evaluation of f

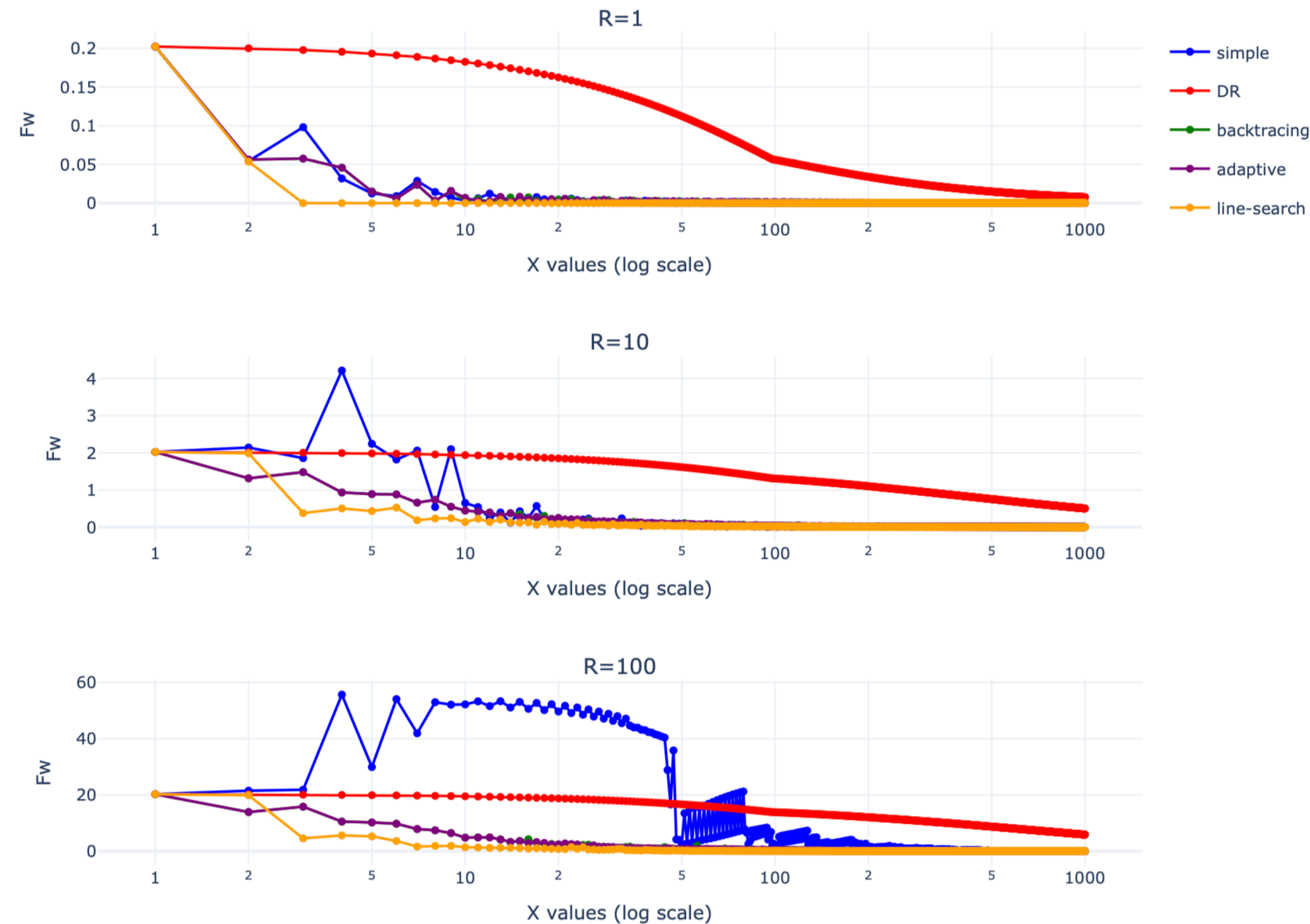
Backtracking line search



Experiments

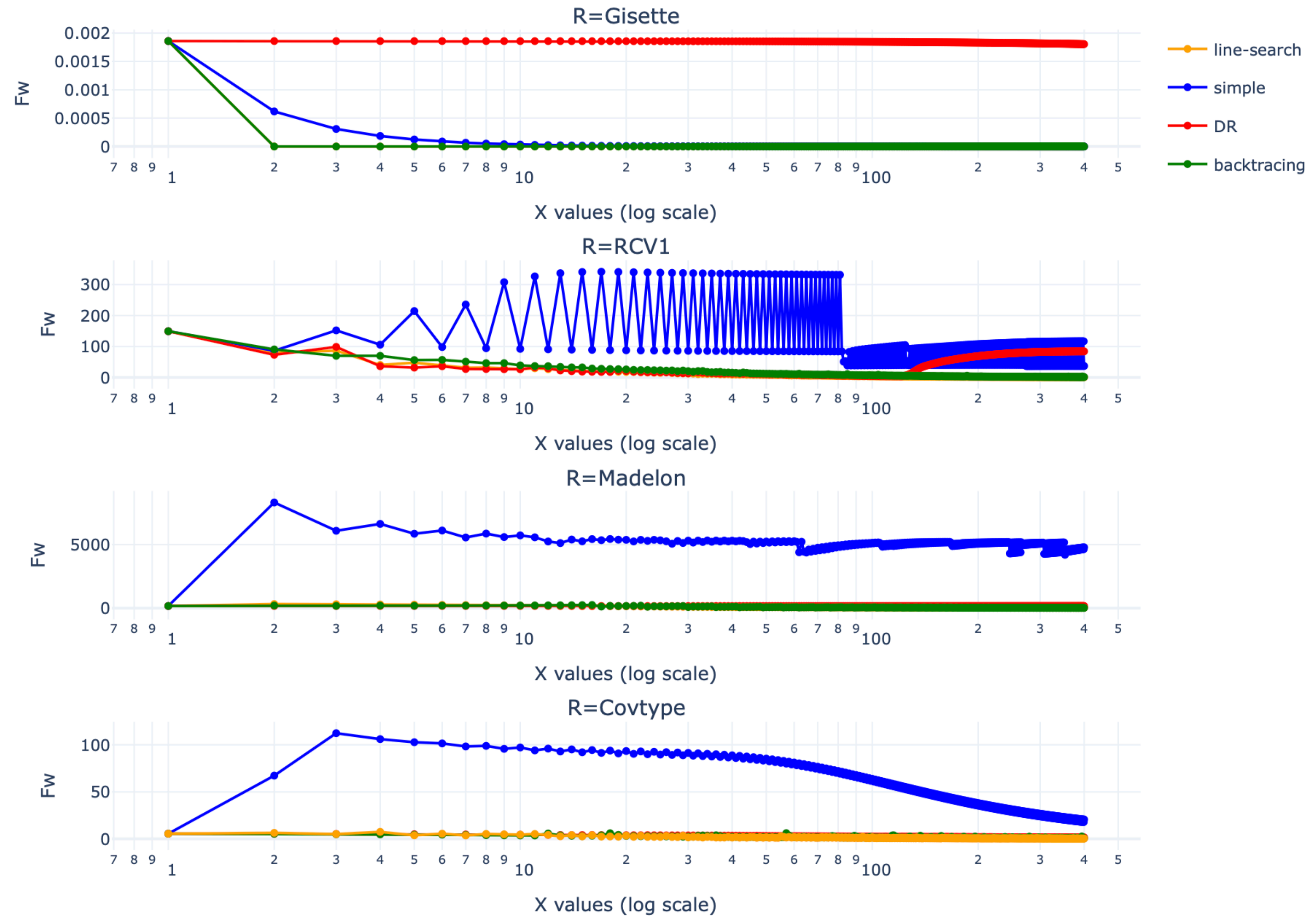
HW (Mushrooms)

Line Plots for Metric: fw (Grouped Legend)



Benchmark

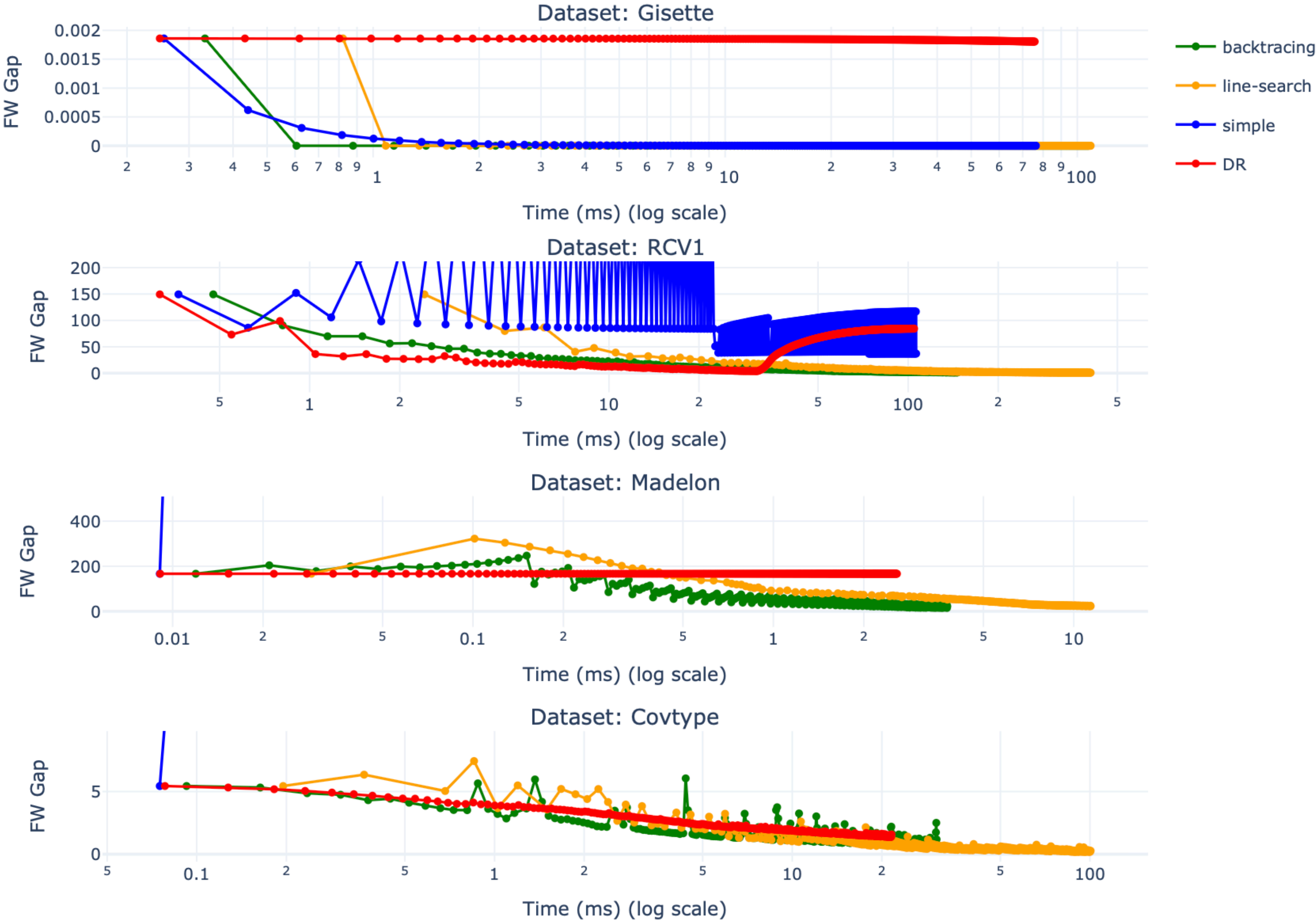
- Gisette: Binary classification on 5000 features $R = 6e^{-3}$
- RCV1: Binary classification on 47236 features $R = 2e^4$
- Madelon: Binary classification on 500 features $R = 20$
- Covtype: Binary classification on 54 features $R = 200$



Performance

	Simple	DR	Backtracking	Line-Search
Gisette	1:16 5.24 it/s	1:15 5.27 it/s	1:49 3.66 it/s	1:49 3.67 it/s
RCV1	1:43 3.87 it/s	1:41 3.94 it/s	2:25 2.75 it/s	6:41 1 it/s
Madelon	0:02 157.22 it/s	0:02 159 it/s	0:03 107 it/s	0:11 35.84 it/s
Covtype	0:21 18.84 it/s	0:21 18.67 it/s	0:30 13.02 it/s	1:39 4.02 it/s

Benchmark. Performance experiment



Results

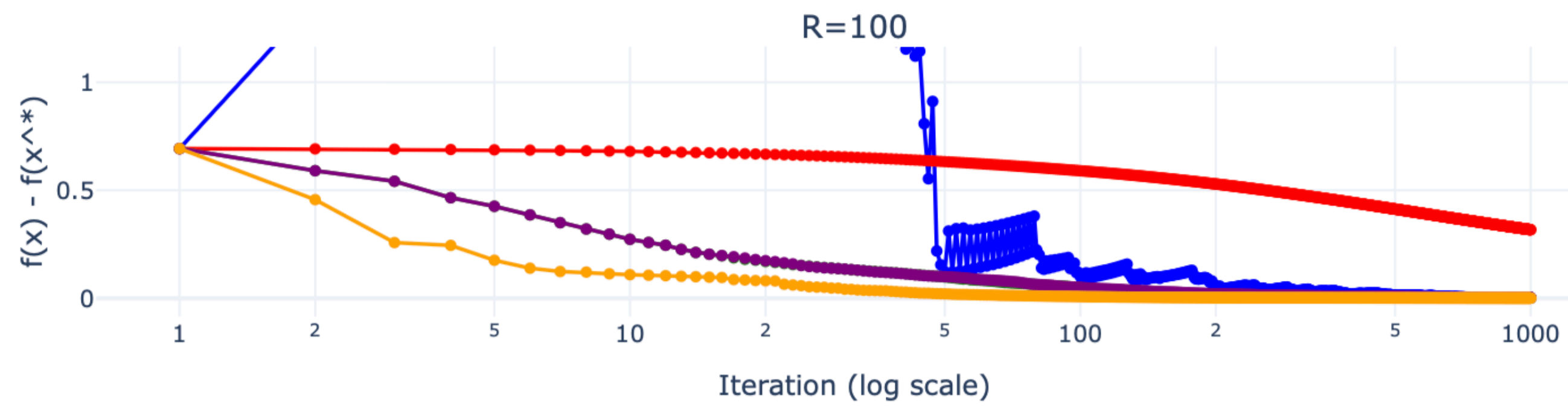
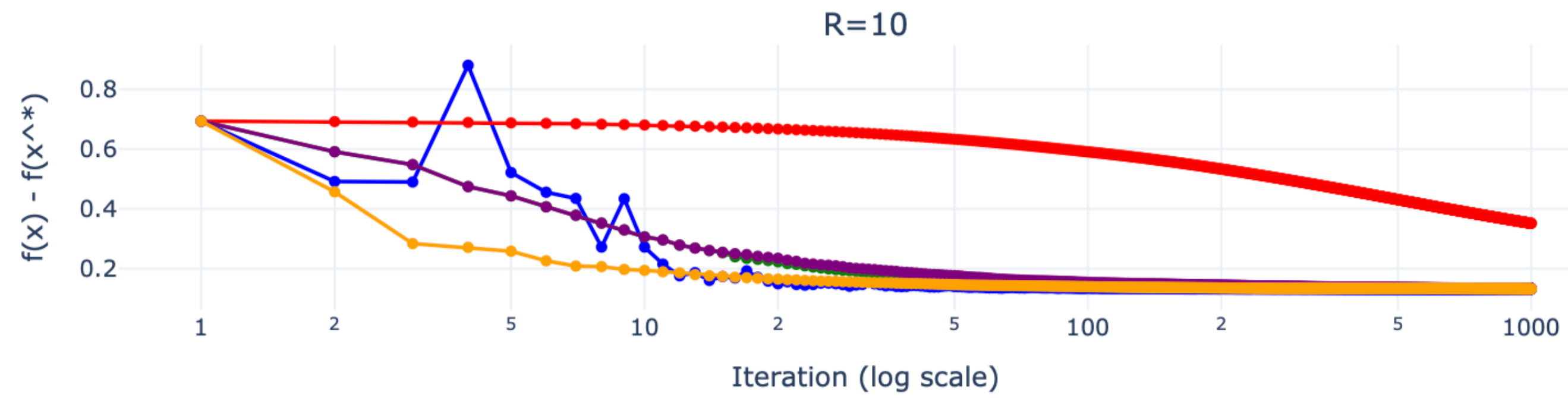
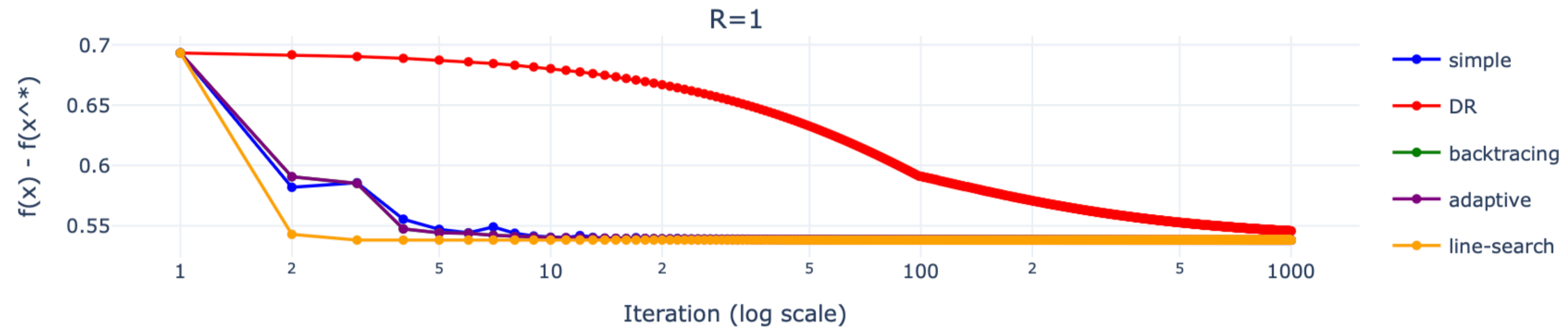
- Linear step: OK
- Demyanov-Rubinov: Sucks
- Backtracking: Best tradeoff
- Line-Search: Best result at cost of high computations

References

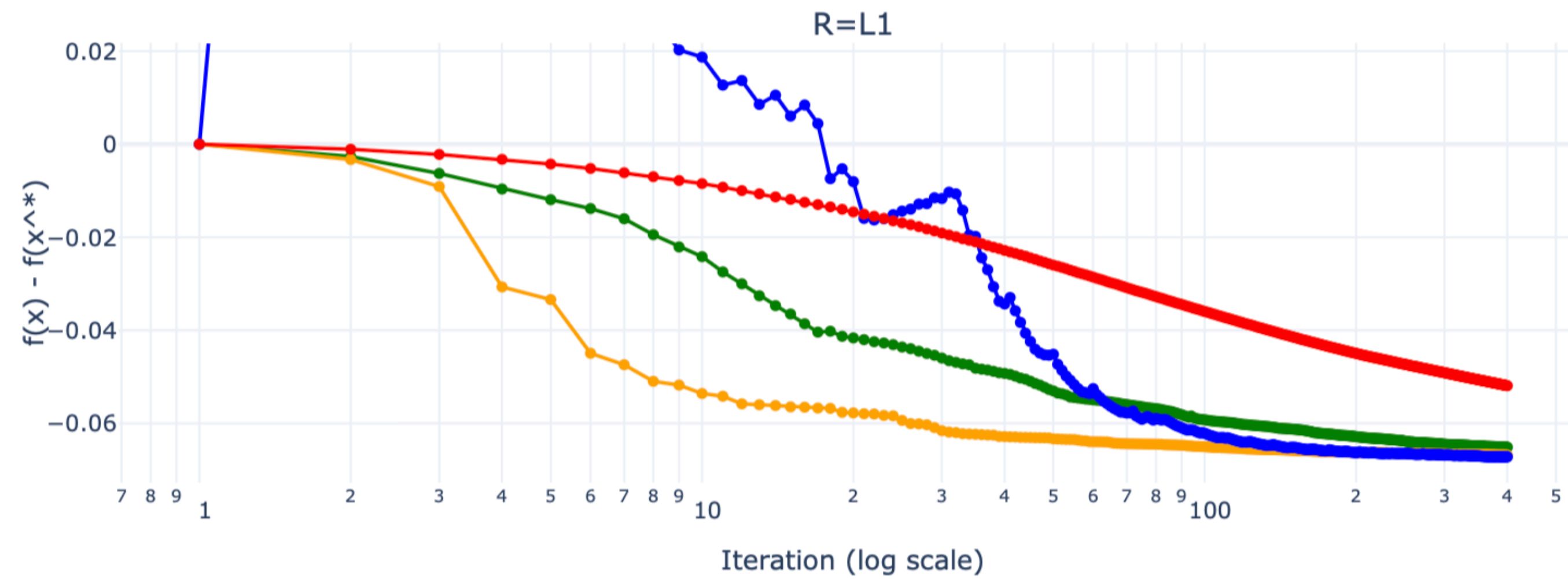
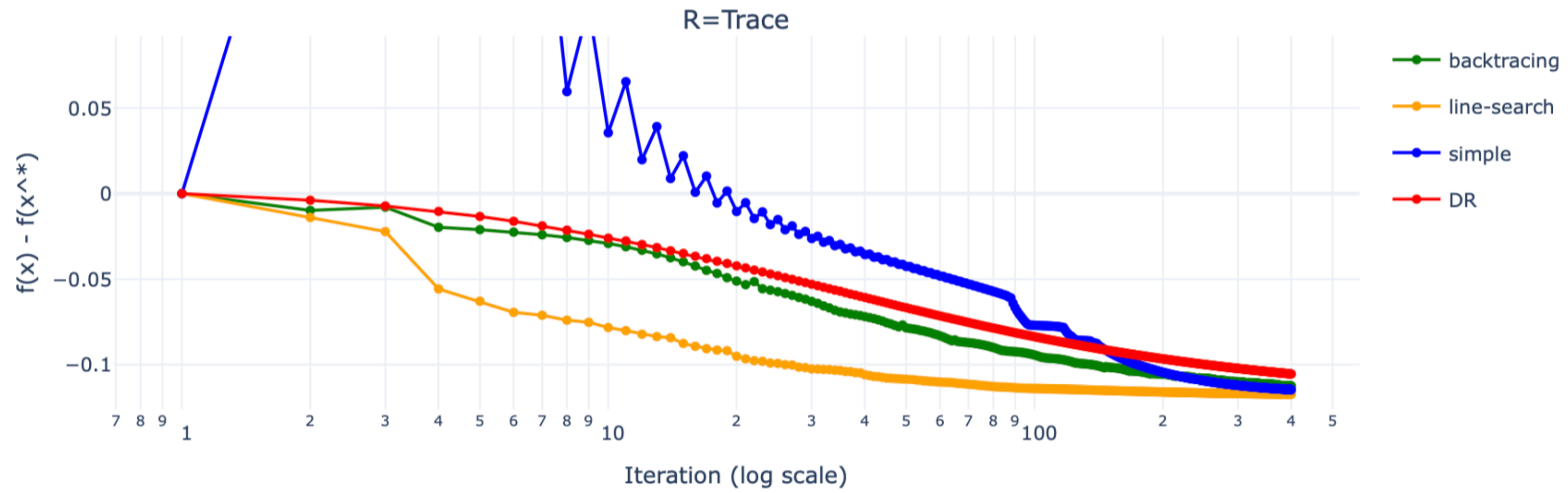
- https://fa.bianp.net/blog/2022/adaptive_fw/
- <https://proceedings.mlr.press/v206/wirth23a/wirth23a.pdf>
- <https://www.researchgate.net/publication/259758847>
- <https://arxiv.org/abs/2311.05313>
- <https://arxiv.org/abs/2002.04756>
- <https://arxiv.org/abs/1511.05932>

Appendix

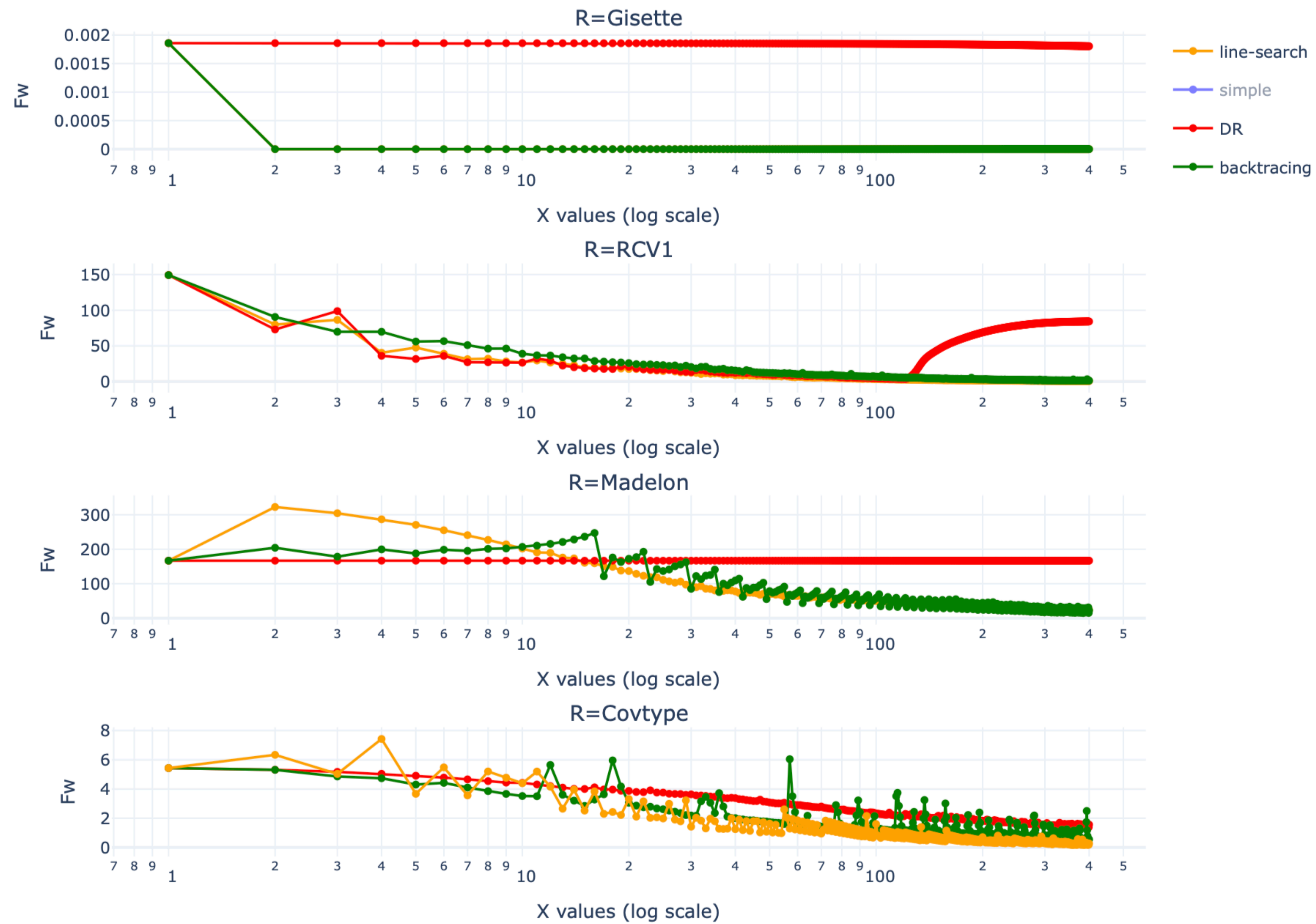
Mushrooms



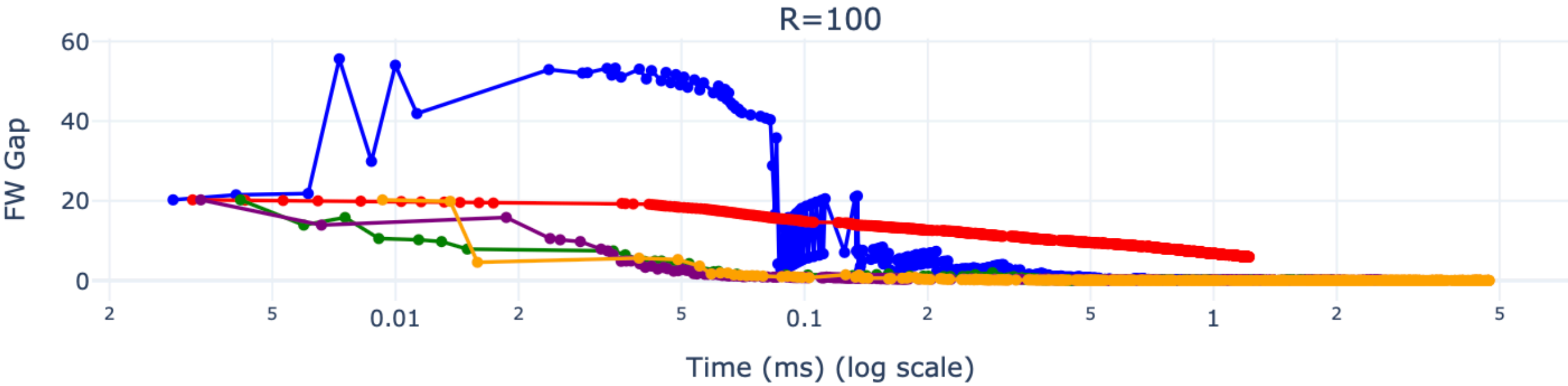
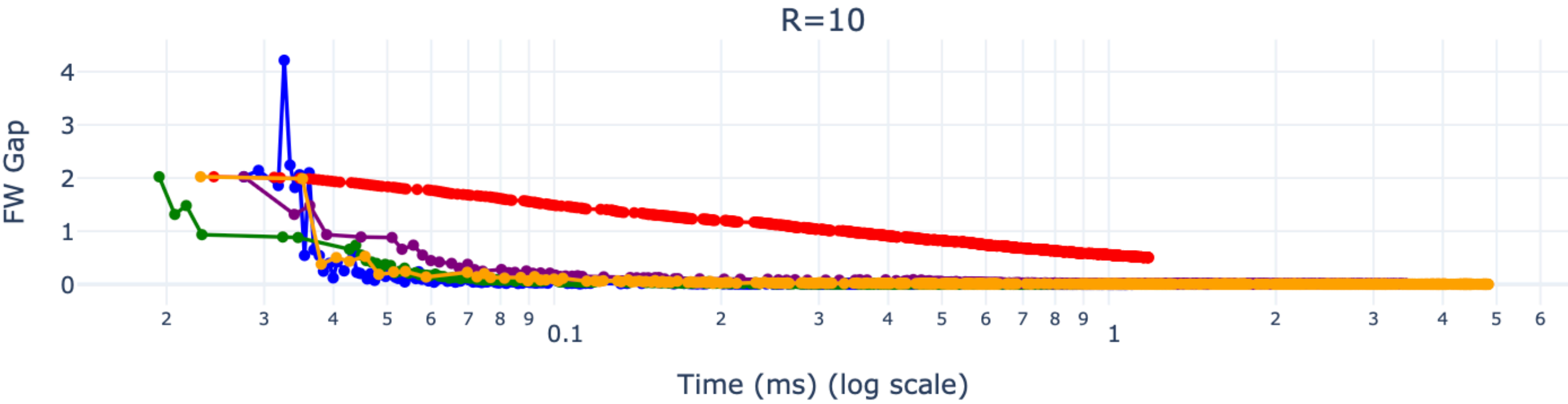
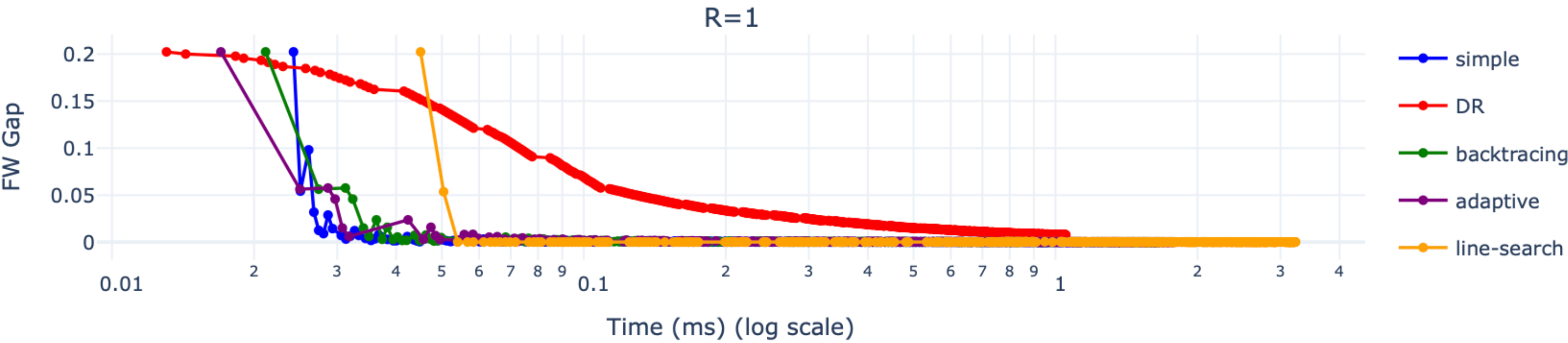
Cancer



Line Plots for Metric: fw (Grouped Legend)



Mushrooms dataset. Performance experiment



Cancer

- Breast Cancer:

Cancer dataset

