The mod-minimizer: a simple and efficient sampling algorithm for long *k*-mers

Giulio Ermanno Pibiri
Ca' Foscari University of Venice

@giulio_pibiri

@jermp

24-th WABI
Egham, UK, 2 September 2024

Joint work with
Ragnar Groot Koerkamp
ETH, Zurich

Sketching with minimizers

- Consider each window of w consecutive k-mers from a string S: sample one k-mer out of w and call it the “representative” of the window — or its *minimizer*.

- We would like to sample the **same minimizer** from consecutive windows so that the **set of distinct minimizers** forms a succinct sketch for S.

- This reduces the memory footprint and comput. time of countless applications in Bioinformatics: such as:
  - sequence comparison,
  - assembly,
  - construction of compacted DBGs,
  - sequence indexing, etc.

Example for $w = 4$ and $k = 7$.

```
ACGGTAGAACCGATTCAAATTCGAT…

ACGGTAGAAC
 CGGTAGAACC
  GGTAGAACCG
   GTAGAACCGA
    TAGAACCGAT
     AGAACCGATT
      GAACCGATTC
       AACCGATTCA
         …
```

Sketching with minimizers

- **Q.** How do we compare different sampling algorithms?

  A. We define the *density* of a sampling algorithm as the fraction between the number of (distinct) minimizers and the total number of k-mers of S.

  The lower the density, the better!

- Since the "window guarantee" must be respected, we immediately have a **lower bound** of $1/w$ on the density of any sampling algorithm.

```
TAGAACCGAT
 AGAACCGATT
  GAACCGATTC
   AACCGATTCA
     ...
```

Example: the “folklore” minimizer

```
function MINIMIZER(W, w, k, O_k)
    o_min = +∞
    p = 0
    for i = 0; i < w; i = i + 1 do
        o = O_k(W[i..i + k))
        if o < o_min then
            o_min = o
            p = i
    return p
```

Example for $w = 4$ and $k = 7$.

ACGGTAGAACCGATTCAAATTCGAT…

ACGGTAGAAC
CGGTAGAACC
GGT**AGAACCG**
GT**AGAACCG**A
TAG**AACCGAT**
AG**AACCGAT**T
G**AACCGAT**TC
AACCGATTCA
…

- We usually define the total order using a random hash function (*random* minimizer).
- In this case, the density is $2/(w+1)$: almost a factor of 2 away from the lower bound for large w.

Introducing the *mod-sampling* algorithm

function MINIMIZER$(W, w, k, \mathcal{O}_k)$

    $o_{min} = +\infty$

    $p = 0$

    for $i = 0;\ i < w;\ i = i + 1$ **do**

        $o = \mathcal{O}_k(W[i..i+k))$

        if $o < o_{min}$ **then**

            $o_{min} = o$

            $p = i$

    return p

function MOD-SAMPLING$(W, w, k, t, \mathcal{O}_t)$

    $o_{min} = +\infty$

    $x = 0$

    for $i = 0;\ i < w + k - t;\ i = i + 1$ **do**

        $o = \mathcal{O}_t(W[i..i+t))$

        if $o < o_{min}$ **then**

            $o_{min} = o$

            $x = i$

    $p = x \bmod w$

    return p

Introducing the *mod-sampling* algorithm

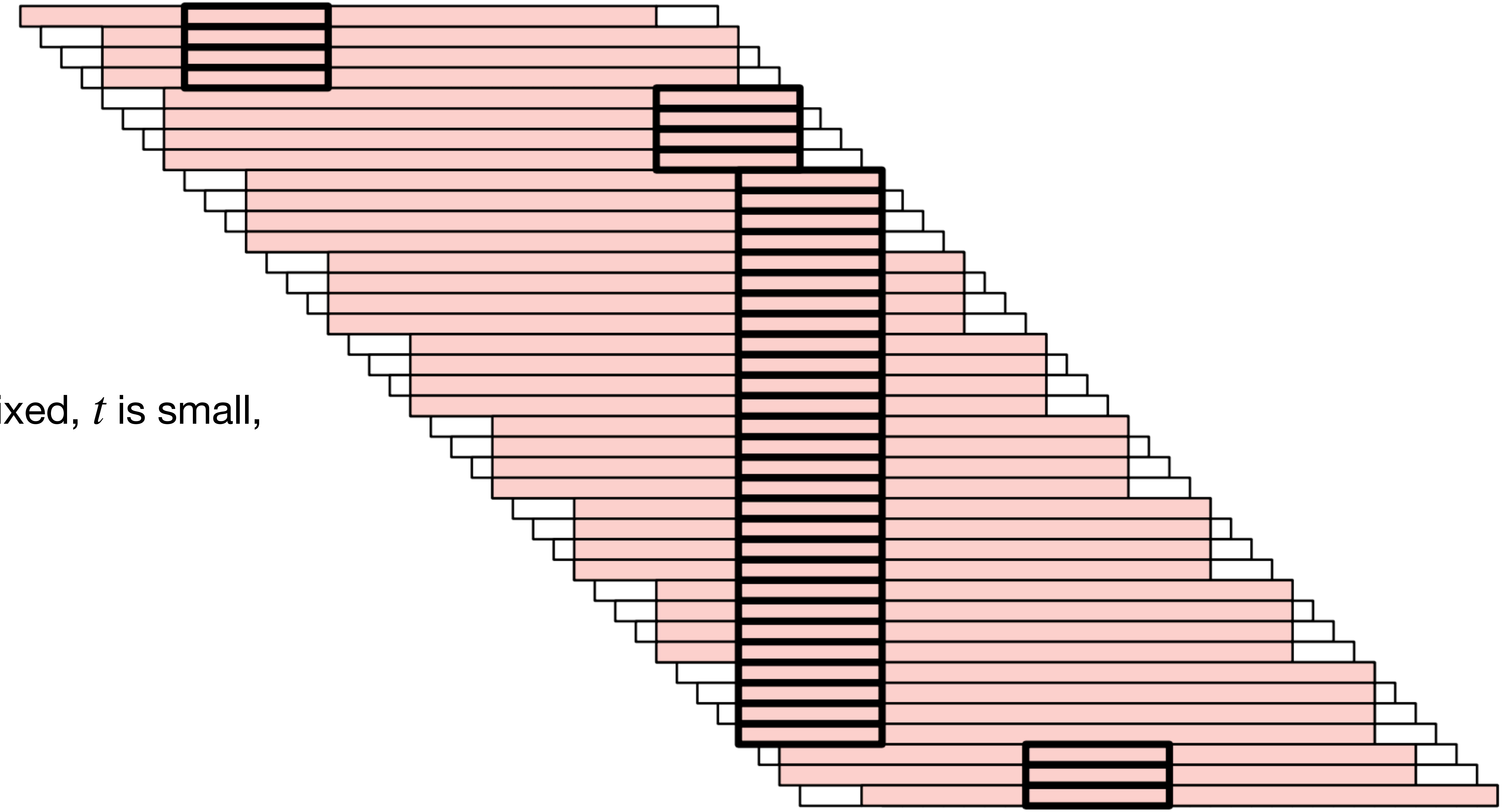
function MINIMIZER$(W, w, k, \mathcal{O}_k)$
  $o_{min} = +\infty$
  $p = 0$
  for $i = 0;\ i < w;\ i = i + 1$ **do**
    $o = \mathcal{O}_k(W[i..i + k])$
    if $o < o_{min}$ **then**
      $o_{min} = o$
      $p = i$
  return p

take smallest k-mer

function MOD-SAMPLING$(W, w, k, t, \mathcal{O}_t)$
  $o_{min} = +\infty$
  $x = 0$
  for $i = 0;\ i < w + k - t;\ i = i + 1$ **do**
    $o = \mathcal{O}_t(W[i..i + t])$
    if $o < o_{min}$ **then**
      $o_{min} = o$
      $x = i$
  $p = x \bmod w$
  return p

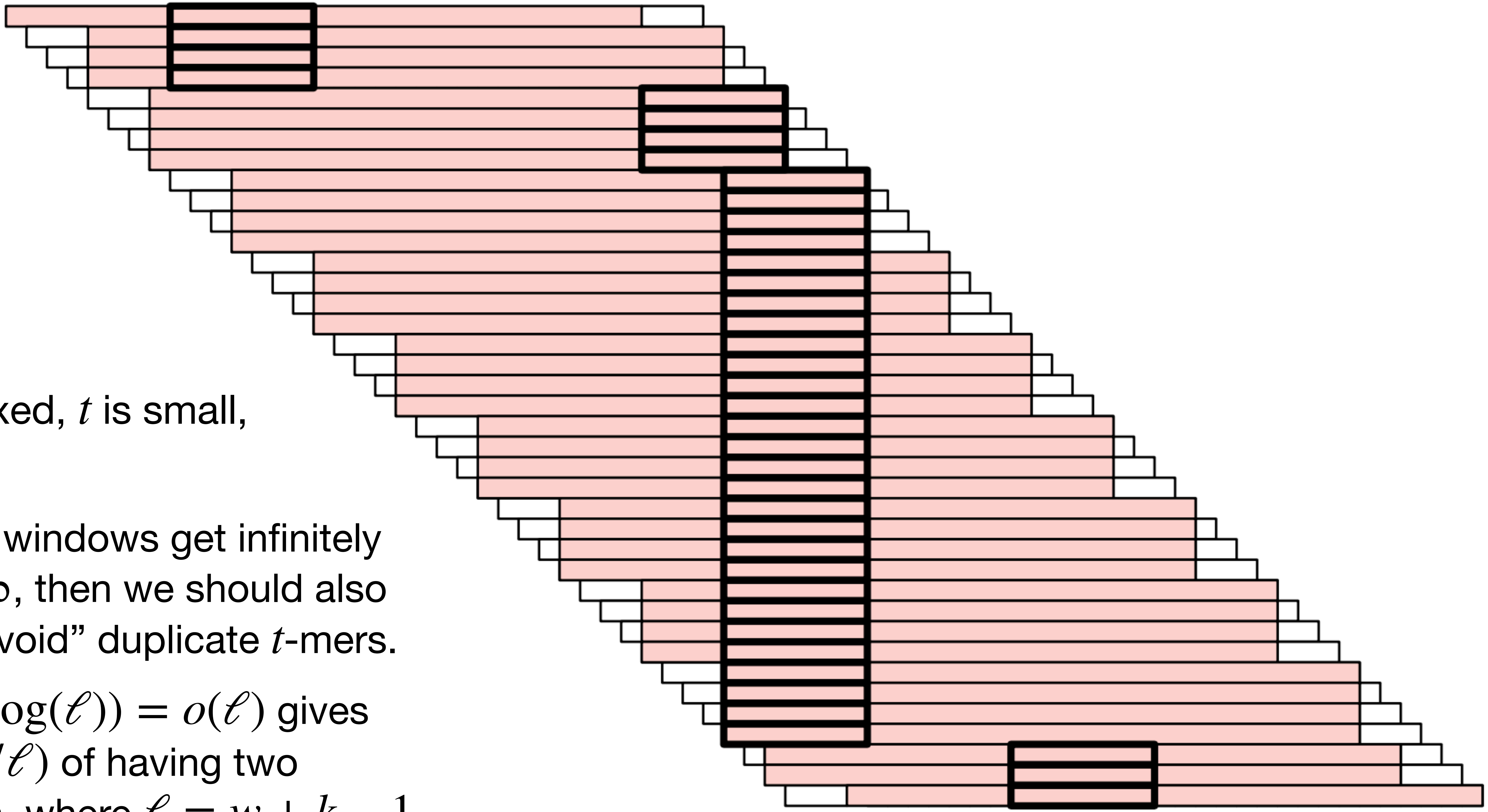
take smallest t-mer,
for some $t < k$

Why does mod-sampling work well for large k?



- Assume w is fixed, t is small, and $k \to \infty$.

Why does mod-sampling work well for large k?



- Assume w is fixed, t is small, and $k \to \infty$.
- One caveat: as windows get infinitely large as $k \to \infty$, then we should also increase t to “avoid” duplicate t-mers.
- Setting $t = \Theta(\log(\ell)) = o(\ell)$ gives probability $o(1/\ell)$ of having two identical t-mers, where $\ell = w + k - 1$.

mod-sampling is optimal for large *k*

- We have a closed-form formula for the density of mod-sampling:

$$\frac{\left\lfloor \frac{\ell - t}{w} \right\rfloor + 2}{\ell - t + 2} + o(1/\ell)$$

where $\ell = w + k - 1$

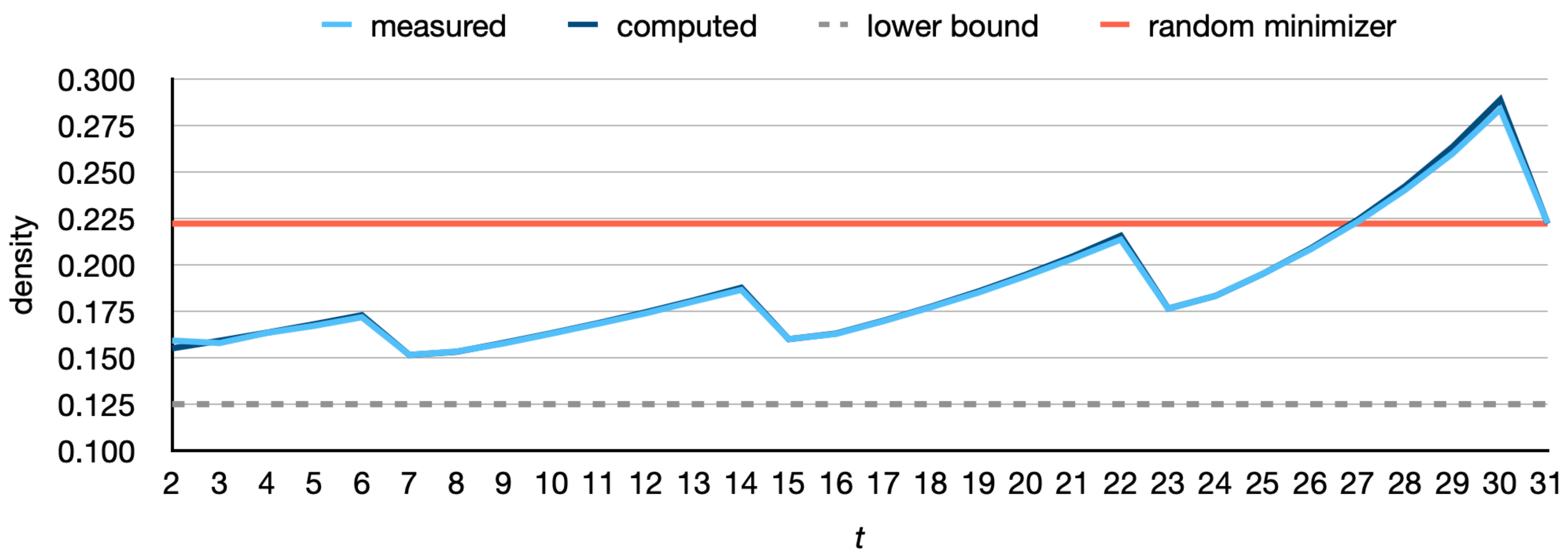
mod-sampling is optimal for large *k*

- We have a closed-form formula for the density of mod-sampling:

$$\frac{\left\lfloor \frac{\ell - t}{w} \right\rfloor + 2}{\ell - t + 2} + o(1/\ell) \xrightarrow[k\to\infty]{} \frac{\frac{\ell - t}{w}}{\ell - t} = 1/w$$

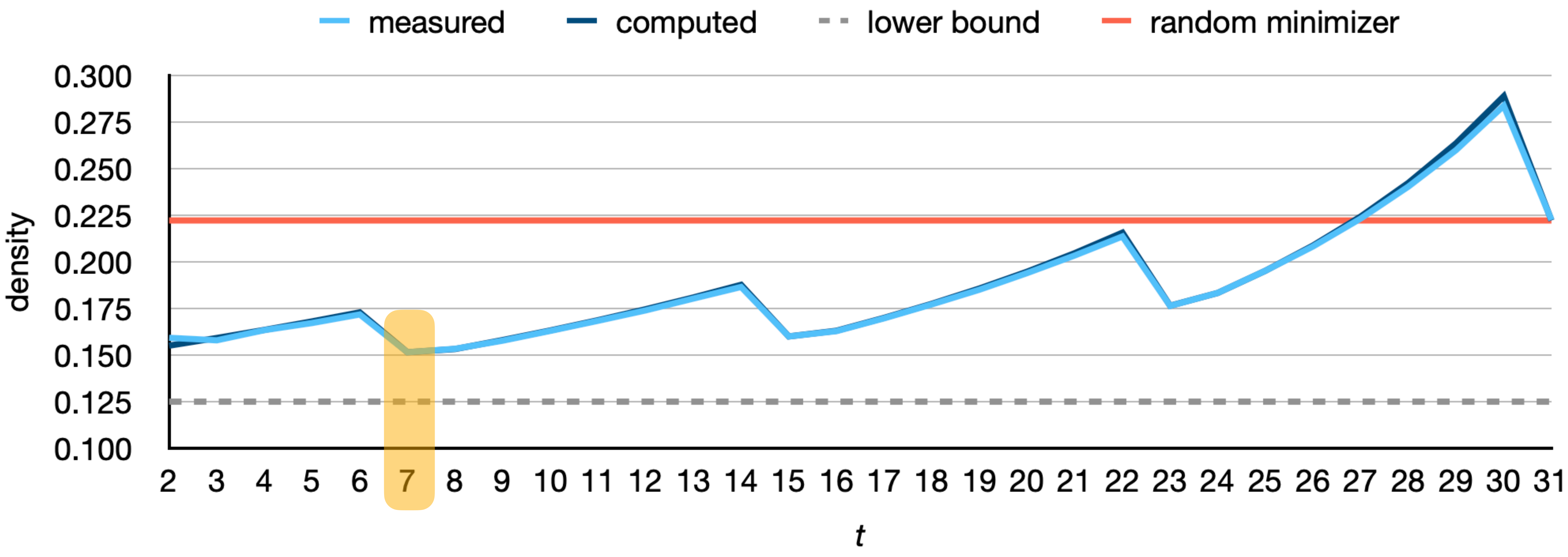
where $\ell = w + k - 1$ (we have $t = o(\ell)$, hence also $\ell - t \to \infty$ as $k \to \infty$)

Density of mod-sampling by varying *t*



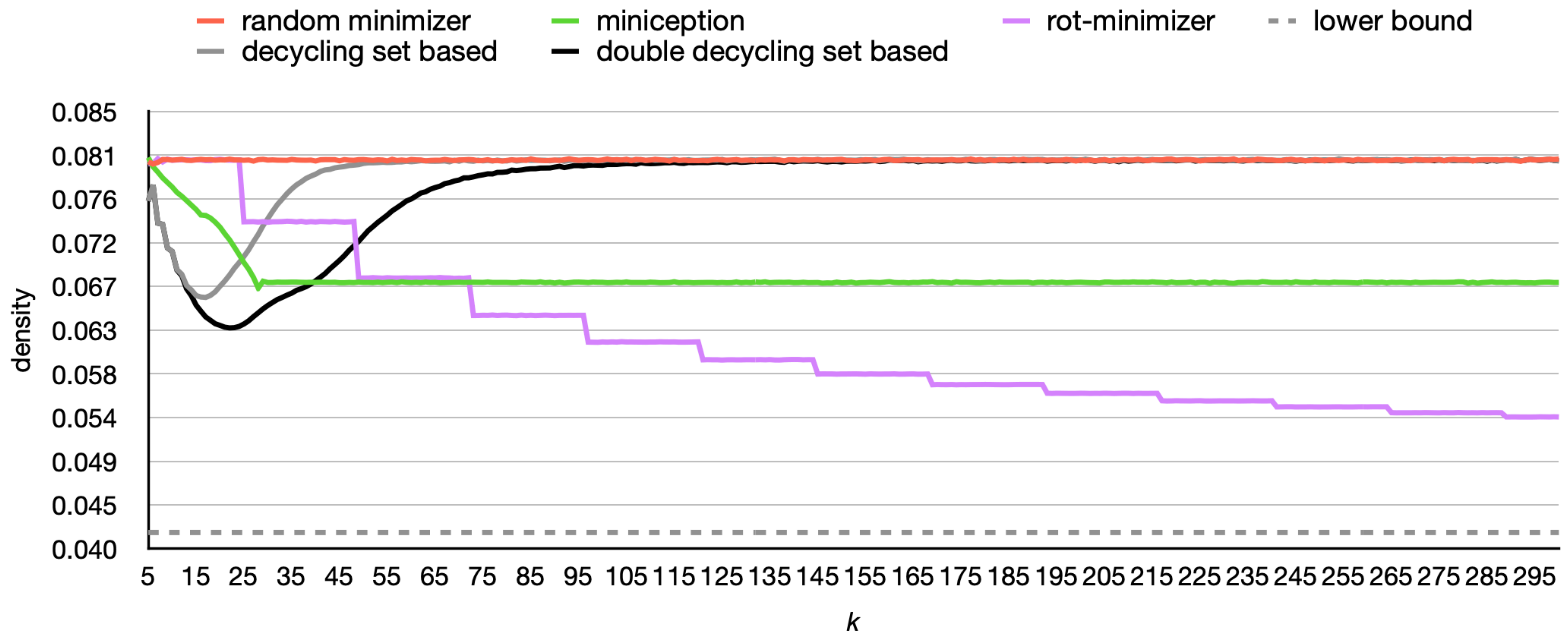
- Example for $k = 31$ and $w = 8$. Measured over a string of 1 million i.i.d. random characters with an alphabet size of 4.

Density of mod-sampling by varying *t*



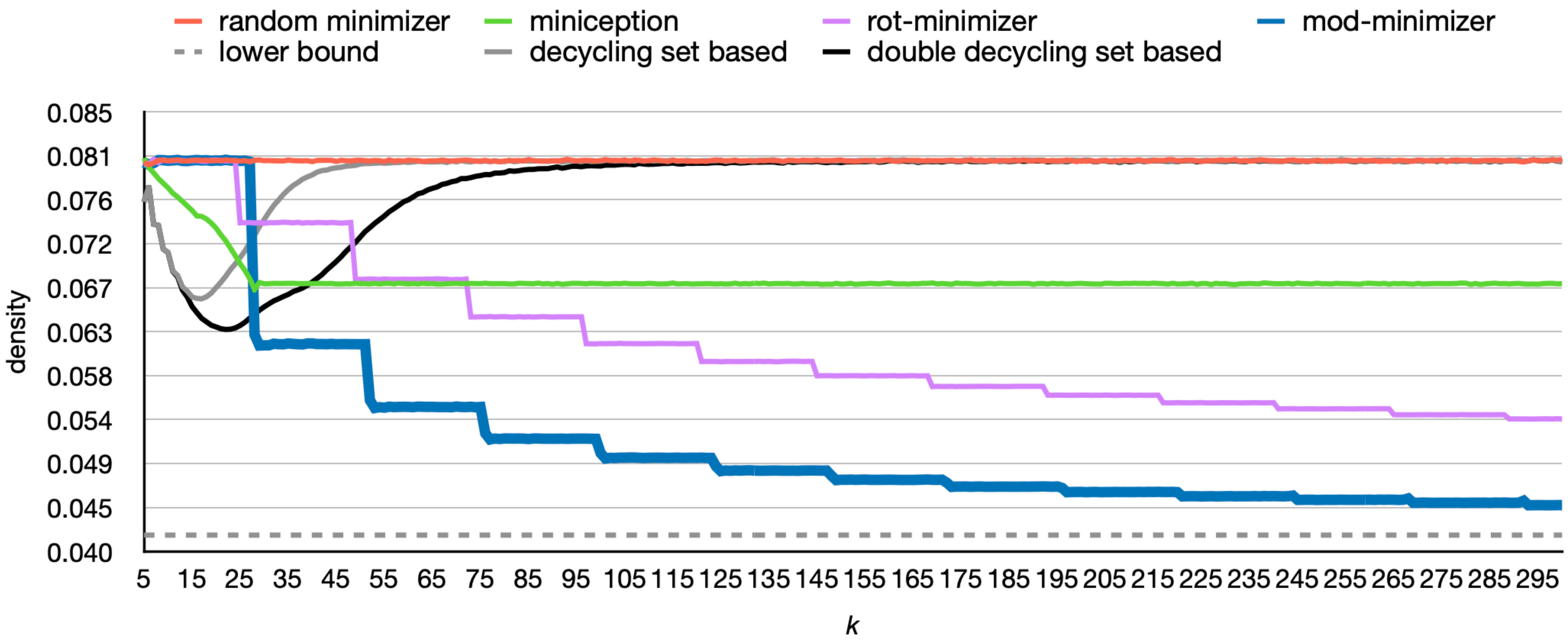
- Example for $k = 31$ and $w = 8$. Measured over a string of 1 million i.i.d. random characters with an alphabet size of 4.
- Density is minimum for the choice $t = k \bmod w$ → **mod-minimizer !**

Density by varying *k*



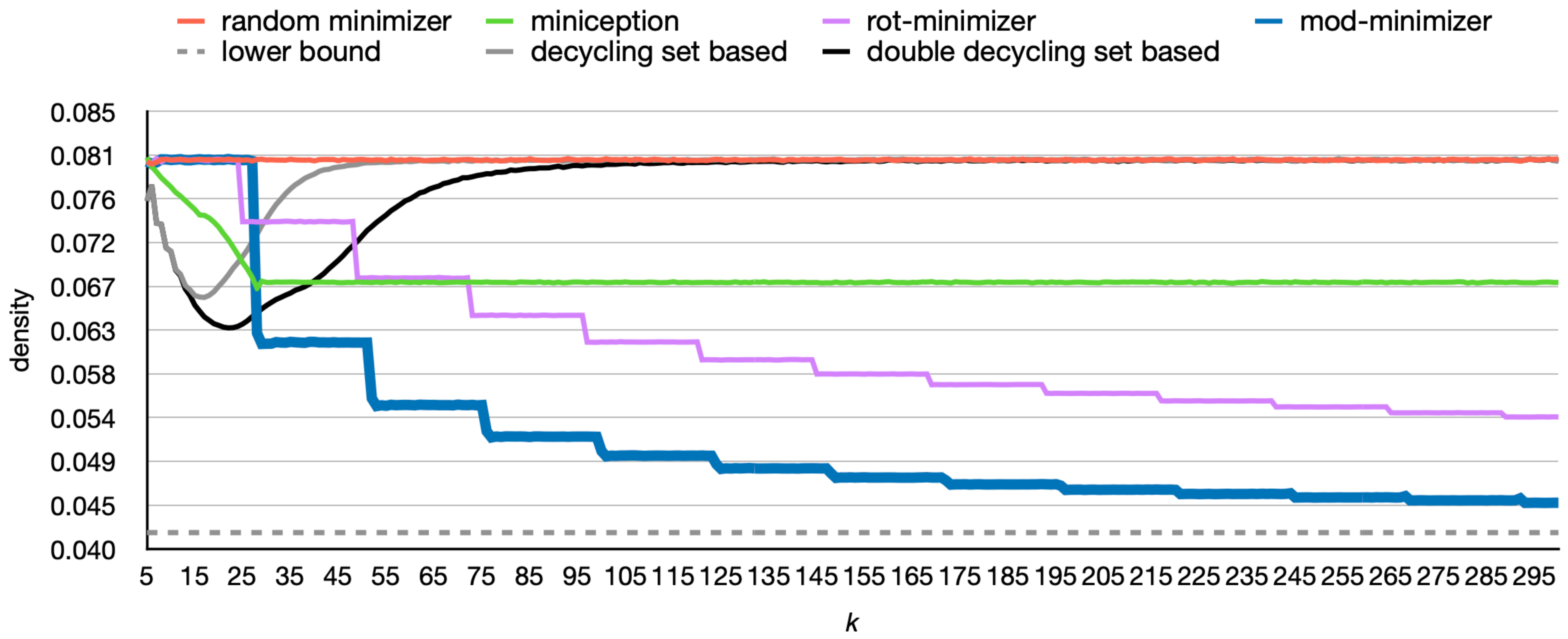
- Example for $w = 24$.
- Measured over a string of 10 million i.i.d. random characters with an alphabet size of 4.

Density by varying *k*



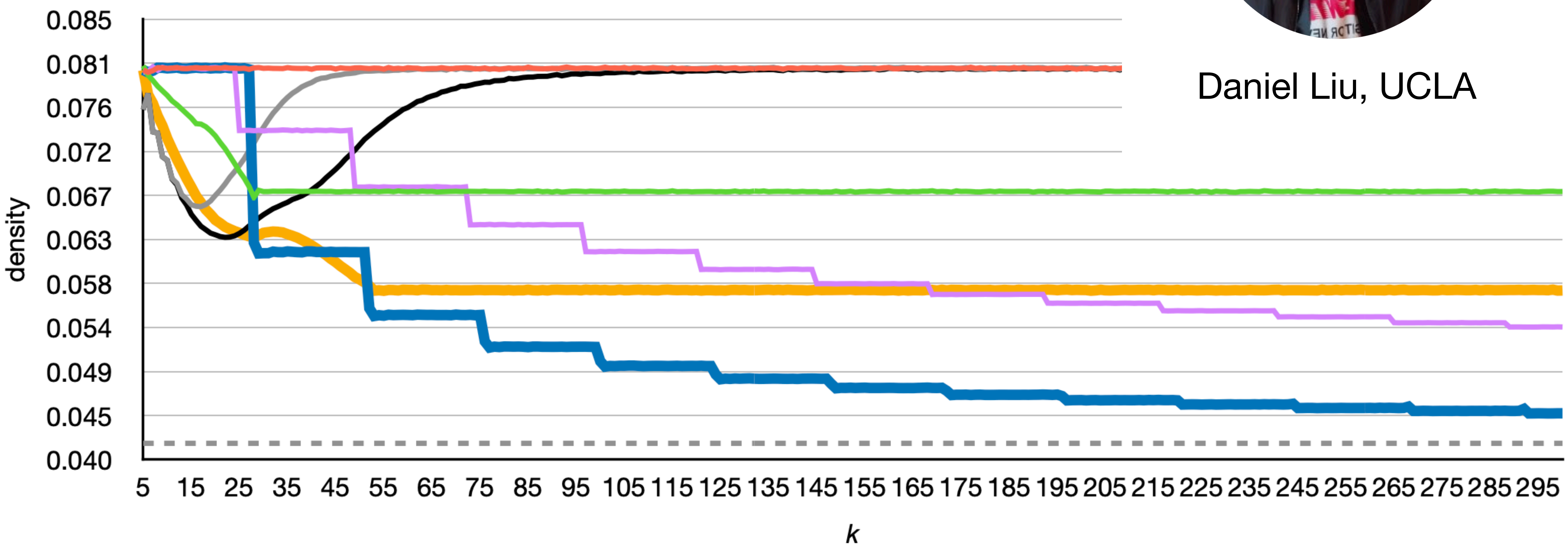
- Example for $w = 24$.
- Measured over a string of 10 million i.i.d. random characters with an alphabet size of 4.

And small *k* ?



- The miniception: sample the **closed syncmer** with the smallest hash value in the window.

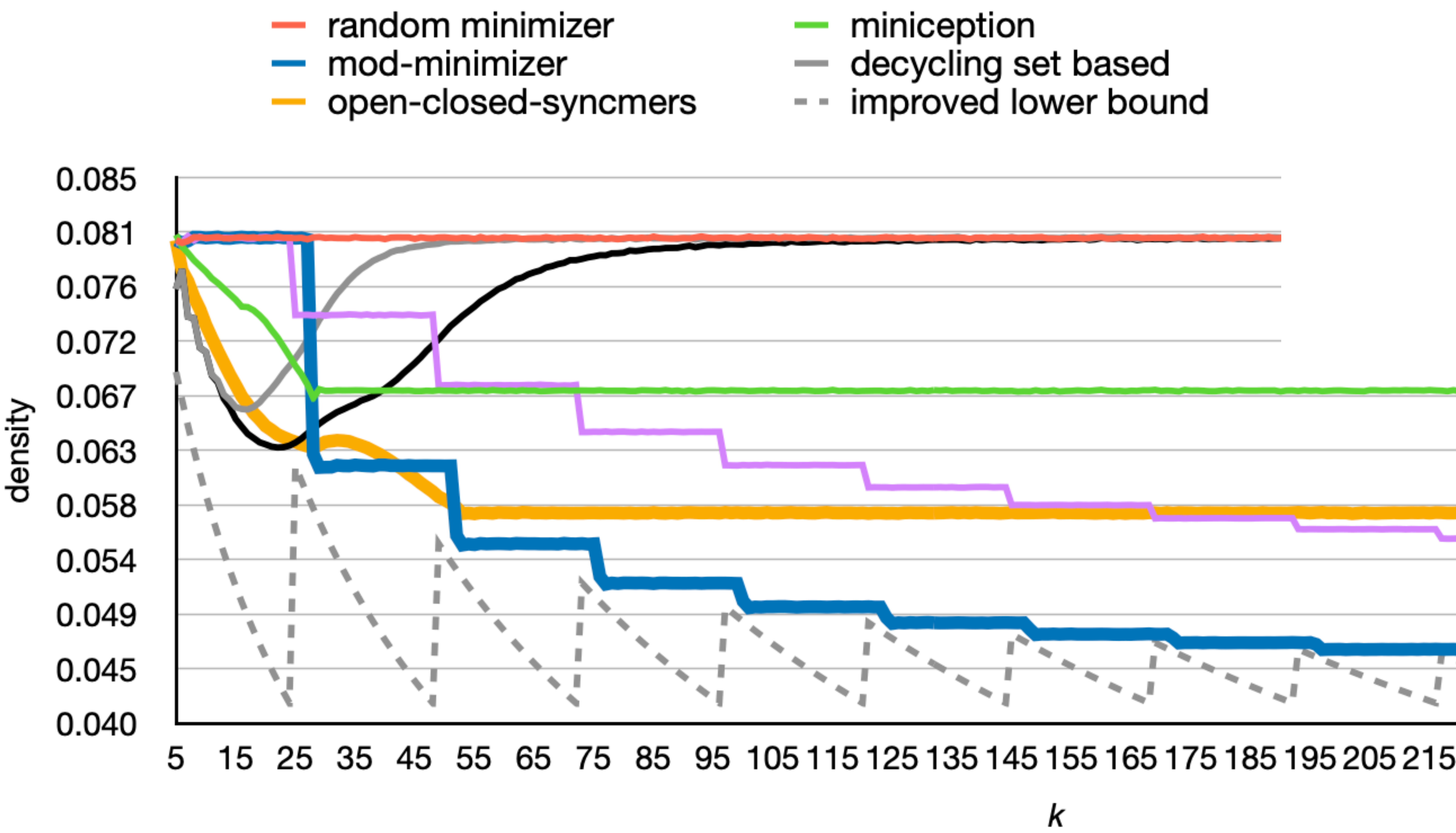
And small *k* ?



Daniel Liu, UCLA

- The miniception: sample the **closed syncmer** with the smallest hash value in the window.
- Daniel: *“ If it works well with closed syncmers, why not trying with **open syncmers** ? ”*

Improved lower bound for small *k*



Bryce Kille
Rice University

- Bryce and Ragnar independently proposed an improved lower bound, which shows that the mod-minimizer is tight when $k \equiv 1 \pmod{w}$.

Conclusions

- We introduced *mod-sampling* — a simple framework that gives new minimizer schemes depending on the choice of a parameter t.
- For $t = k \bmod w$, mod-sampling yields the mod-minimizer that is optimal for $k \to \infty$.
- Replacing random minimizers with mod-minimizers in **SSHash** decreases index space consistently by $\approx$**15%**.
- C++ code: https://github.com/jermp/minimizers
- Rust code: https://github.com/RagnarGrootKoerkamp/minimizers

Conclusions

- We introduced *mod-sampling* — a simple framework that gives new minimizer schemes depending on the choice of a parameter t.
- For $t = k \bmod w$, mod-sampling yields the mod-minimizer that is optimal for $k \to \infty$.
- Replacing random minimizers with mod-minimizers in **SSHash** decreases index space consistently by $\approx$**15%**.
- C++ code: https://github.com/jermp/minimizers
- Rust code: https://github.com/RagnarGrootKoerkamp/minimizers

Thank you for the attention!