Minimizer Density revisited: Models and Multiminimizers

Florian Ingels[1], Lucas Robidou[1,2], Igor Martayan[1], Camille Marchet[1], and Antoine Limasset[1]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2] I2BC, Université Paris-Saclay, CNRS, CEA, Gif sur Yvette, France
antoine.limasset@univ-lille.fr

Abstract. High-throughput sequence analysis commonly relies on k-mers (words of fixed length k) to remain tractable at modern scales. These k-mer-based pipelines can employ a sampling step, which in turn allows grouping consecutive k-mers into larger strings to improve data locality. Although other sampling strategies exist, local schemes have become standard: such schemes map each k-mer to the position of one of its characters. A key performance measure of these schemes is their density, defined as the expected fraction of selected positions. The most widely used local scheme is the minimizer scheme: given an integer $m \leq k$, a minimizer scheme associates each k-mer to the starting position of one of its m-mers, called its *minimizer*. Being a local scheme, the minimizer scheme guarantees covering all k-mers of a sequence, with a maximal distance between selected positions of $w = k - m + 1$. Recent works have established near-tight lower bounds on achievable density under standard assumptions for local schemes, and state-of-the-art schemes now operate close to these limits, suggesting that further improvements under the classical notion of density will face diminishing returns. Hence, in this work, we aim to revisit the notion of density and broaden its scope.

As a first contribution, we draw a link between density and the distance between consecutive selected positions. We propose a probabilistic model allowing us to establish that the density of a local scheme is exactly the inverse of the expected distance between the positions it selects, under the minimal and only assumption that said distances are somehow equally distributed. We emphasize here that our model makes no assumptions about how positions are selected, unlike the classical models in the literature. Our result introduces a novel method for computing the density of a local scheme, extending beyond classical settings.

Based on this analysis, we introduce a novel technique, named multiminimizers, by associating each k-mer with a bounded set of candidate minimizers rather than a single one. The candidate furthest away (in a precise sense defined in the article) is selected. Since the decision is made by taking advantage of a context beyond a single k-mer, this technique is not a local scheme — and belong to a novel category of meta schemes. Using the multiminimizer trick on a local scheme reduces its density at the expense of a controlled increase in computation time. We show that this method, when applied to random (hash-based) minimizers and to open-closed mod-minimizers, approaches a density of $\frac{1}{w}$, representing, to our knowledge, the first construction converging to this limit.

Our third contribution is the introduction of the *deduplicated density*, which measures the fraction of distinct minimizers used to cover all k-mers of a set of sequences. While this problem has gained traction in applications such as assembly, filtering, and pattern matching, standard minimizer schemes are often used as a proxy, blurring the distinction between the two objectives (minimizing the number of selected positions or the number of selected minimizers). Although related to the classical notion of density, deduplicated density differs in both definition and suitable constructions, and must be analyzed in its own right, together with its precise connections to standard density. We show that multiminimizers can also improve this metric, but that globally minimizing deduplicated density in this setting is NP-complete, and we instead propose a local heuristic with strong empirical behavior.

Finally, we show that multiminimizers can be computed efficiently, and provide a SIMD-accelerated Rust implementation together with proofs of concept demonstrating reduced memory footprints on core sequence-analysis tasks. We conclude with open theoretical and practical questions that remain to be addressed in the area of density.

Keywords: density, minimizers, sampling, k-mers, selection scheme, sequence analysis

1 Introduction

Efficient sequence comparison is a central computational pillar of modern genomics. Modern applications must process genomes of hundreds of gigabases, handle terabase-scale sequencing datasets, and search petabase-scale reference collections with both speed and tight memory usage. To perform sequence alignment at large scales, heuristic strategies such as seed-and-extend and chaining [1,2] avoid full dynamic-programming alignment, and prefer dividing this procedure into smaller or simpler tasks for which exact k-mer (a word of fixed length k) matches serve as high-confidence landmarks that guide the alignment. A key observation is that, in practice, one does not need to index all k-mers to maintain high alignment sensitivity; therefore, k-mer sampling is often preferred to decrease resource usage [3]. However, naive sampling schemes lack robustness to mutations or noise, and sampling schemes designed for distance approximation can generate "deserts" where no k-mer is chosen, leaving the alignment algorithm without the necessary anchor points [4]. Minimizers were introduced in this context [5,6] to couple sparsity (through sampling) with coverage guarantees (to avoid deserts as much as possible). This paper addresses several fundamental aspects of minimizers and their effectiveness in sequence representation.

Given a window of $w = k-m+1$ consecutive m-mers ($k \geq m$), a minimizer scheme selects one m-mer in this window according to an order, typically induced by a hash function. This ensures that any exact match of length at least $k = w + m - 1$ between two sequences yields at least one common minimizer, provided they use the same ordering. At first glance, this guarantees coverage, but not efficient sampling, since in principle each overlapping window could select a different m-mer. In practice, successive windows share $w - 1$ m-mers and differ by only one entering and one leaving m-mer, so they tend to select the same minimizer over long stretches. To formalize this, the notion of density was introduced: the average number of selected m-mers per window, or equivalently, the fraction of m-mers retained. This paper focuses on minimizer density, in particular in its optimization. To be explicit, a lower density means that fewer m-mers are selected, which directly reduces memory usage and comparison cost, or expectedly more cache-coherent algorithms, making it both a theoretically interesting question and one with broad practical impact, described succinctly below.

A wide range of tools now rely on minimizers as their main alignment and indexing mechanism. Long-read [7,8] and whole-genome aligners [9] index only minimizers for their heuristics, reducing memory footprint and accelerating chaining. Classifiers and document-retrieval indexes use minimizers to shrink database size by orders of magnitude [10] or to speed up filtering [11,12]. Lossy representations encoding sequences as ordered lists of minimizers have also become central in lightweight assembly and comparison methods [13,14]. Beyond window-based selection, minimizer schemes can assign to each k-mer one of its m-mer ($m \leq k$) as its minimizer, inducing a natural partition of the k-mers without additional metadata. Consecutive k-mers often share the same minimizer, forming long runs that can be stored compactly as a single superstring of length $k+m-1$ [15]. This property is exploited in k-mer counters [16,17], scalable de Bruijn graph construction [18,19,20], and large-scale indexing structures [21,22]. In this setup, the density matters again: higher density breaks these runs more often, reducing the benefits of the compact string representation.

Minimizers are typically computed on a stream of genomic sequences. A first key notion is that of local schemes, in which m-mers are selected solely based on the content of each window. A second notion is that of forward schemes, which restrict selection to m-mers appearing at increasing positions along the sequence. Every forward scheme is therefore a local scheme. With this window structure in place, the best possible density is $\frac{1}{w}$ (one selected k-mer per window). Early work established random-order minimizers, e.g. hashing-based orderings, as a simple baseline with expected density $\frac{2}{w+1}$, and conjectured that this might be near-optimal for local schemes [5]. Subsequent results disproved this, and multiple research threads have since pushed densities closer to the theoretical limit:

- Marçais et al. characterized when asymptotically optimal minimizers exist and introduced a practical construction [23].
- Another line of work proposed precomputing a set of minimizers that hits every possible window in the universe; so that a given input sequence then only selects minimizers from this set. These schemes improve density but suffer from exponential growth in construction cost as k increases. Pellow et al. addressed this with heuristics that scale to arbitrarily large k and any window length [24].
- Groot Koerkamp and Pibiri [25,26] introduced schemes based on modular constraints on internal subwords, achieving asymptotically optimal density $\frac{1}{w}$ as $k \to \infty$.

Kille and Groot Koerkamp et al. later derived a nearly tight lower bound for forward sampling schemes, showing that for many practical (k, w), state-of-the-art constructions are within only a few percent of optimal density [27]. This suggests that substantial further improvements in local schemes are unlikely.

Recent schemes such as Golan et al. [28] attain densities extremely close to the Kille and Groot Koerkamp et al. bound for practical parameter ranges, as do the aforementioned constructions of Pellow et al., and Groot Koerkamp and Pibiri.

To escape the limitations of universal forward local schemes, which are designed to perform well on any input, the community has also designed sequence-specific schemes that tailor minimizer selection to the properties of a given genome. These include frequency-based selection [29], spacing-based strategies [30], and machine-learning approaches [31]. While these methods can substantially reduce density on real genomes, they cannot be computed in a streaming fashion, making them impractical at very large scales.

Despite the extensive effort devoted to low-density minimizer schemes, several aspects of density behavior remain poorly understood. In this work, we focus on three such aspects and use them as the foundation for our contributions.

First, we revisit classical density for local schemes in a unified way. We formalize the link between density and distances between consecutive selected positions and show that, under a minimal assumption on these distances, the density of *any* local scheme equals the inverse of the expected distance. Specializing to random minimizers, we argue that the standard modeling assumption is unsatisfactory. We thus propose a more realistic one that coincides with the classical regime for large m-mers, and verify empirically that it satisfies our minimal condition.

Second, we build on this perspective to design practical family of meta schemes dubbed *multiminimizers*. Instead of fixing a single minimizer per k-mer, we assign a bounded set of candidate minimizers and choose among them to reduce the frequency of selected positions while preserving locality and consistency. Instantiated on top of a random-hash order, multiminimizers attain asymptotically optimal positional density, match or surpass the best known schemes in practice, and remain simple and efficient to implement.

Third, we make explicit a twin notion to classical density: the *deduplicated density*, the fraction of distinct minimizers needed to cover all k-mers, directly relevant for filters and indexes. We show that classical and deduplicated densities diverge on long random sequences, but essentially coincide for small sequences, when computed for random minimizers. We then study minimizing deduplicated density in the multiminimizer framework, prove that the global optimization problem is NP-complete (via reduction from set cover), and introduce a local heuristic with strong empirical behavior. This detour clarifies the relationship between the two densities and highlights a remaining open problem: establishing the computational complexity of minimizing classical density itself.

2 Methods

2.1 Preliminaries

In this section, we introduce key concepts for the rest of this article. We build mainly on the notations of [17,32,25,33,28]. We are interested in strings defined over the DNA alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ of size $\sigma = 4$. Let $S \in \Sigma^*$ be a string; we denote by $S[i \dots j)$ the substring of S of length $j - i$, starting at position i (included) and ending at position j (excluded). A k-mer is a substring $S[i \dots i+k)$ of fixed length k. The length of a string S is denoted by $|S|$. Most of the time, the DNA strand from which a sequence is coming is unknown; the ambiguity is resolved by considering that a k-mer and its reverse complement are, in fact, the same k-mer. One of the two (e.g. the lexicographically smallest) is chosen to be the *canonical* representative of the pair. Hereafter, the definitions are stated in a non-canonical context (but are easily extended); whereas the practical experiments presented later on are based on canonical k-mers, although our implementations cover both contexts. On the topic of canonical k-mers, see also [4].

An order $\mathcal{O}_m$ on m-mers is an injective function $\Sigma^m \to \mathbb{R}$ such that $x \leq_{\mathcal{O}_m} y \iff \mathcal{O}_m(x) \leq \mathcal{O}_m(y)$. In practice, we define an order using a (pseudo)-random hash function $h : \Sigma^m \to U$ [34], where U is large enough so that the probability of collision (i.e. $h(x) = h(y)$ with $x \neq y$) can be assumed to be 0 – i.e. h is a perfect hash function. Typically, $U = 2^{64}$ whenever using 64-bits hash functions. We denote by $\mathtt{rank}_{\mathcal{O}_m} : \Sigma^m \to [\![1, \sigma^m]\!]$ the function that associates to a m-mer its rank among all m-mers according to the order $\mathcal{O}_m$. Since $\mathcal{O}_m$ is injective, $\mathtt{rank}_{\mathcal{O}_m}$ is a bijection.

Definition 1 ((Random) Minimizer). *The minimizer* $\min(x)$ *of a k-mer x is its smallest substring of size $m < k$ for the (random) order $\mathcal{O}_m$.*

Let us denote $w = k - m + 1$ the number of m-mers contained in a k-mer.

Definition 2 (Minimizer scheme). *Provided an order $\mathcal{O}_m$, a minimizer scheme is a function $f : \Sigma^k \to [\![1, w]\!]$ that returns the position of the minimizer of a k-mer, with ties resolved to the left; i.e. $f(x) = \min\{1 \leq i \leq w : x[i \dots i + m) = \min(x)\}$.*

Definition 3 (Set of selected indices). *The set of selected indices of a string S using the minimizer scheme f is defined as $\mathcal{S}_f(S) = \{i + f(S[i \dots i+k)) : 1 \leq i \leq |S| - k + 1\}$.*

As noted in [33], the scheme may select the same position in consecutive windows (since consecutive windows overlap by $k-1$ bases), so that $|\mathcal{S}_f(S)| \leq |S| - k + 1$.

Definition 4 (Density, [28]). *The (expected) density of a minimizer scheme f is formally defined as*

$$d = \lim_{M \to \infty} \frac{1}{\sigma^M} \sum_{S \in \Sigma^M} \frac{|\mathcal{S}_f(S)|}{|S| - k + 1} \tag{1}$$

and represents the expected proportion of selected minimizers among all m-mers on a random string, where each character is chosen uniformly at random. For any sequence S, the term $\frac{|\mathcal{S}_f(S)|}{|S|-k+1}$ is also defined as the particular density of S, and denoted by $d(S)$. Since $1/w \leq d(S) \leq 1$, we also have $1/w \leq d \leq 1$.

Marçais et al. proved that the density can be computed exactly (instead of just being estimated) by noticing that the density and the particular density coincided when computed on a de Bruijn sequence of large enough order [35]. The purpose of the upcoming Section 2.2 is to provide yet another way to compute the density. Finally, note that all these definitions can also be applied to the broader concept of *local schemes*, which are guaranteed to select one m-mer per k-mer. Minimizer schemes are one example of local schemes, see also [33].

2.2 A link between density and the expected distance between selected positions

Let f be a minimizer scheme. Consider the expected number of selected positions in a window of size w — by definition, is is equal to $d \cdot w$. Intuitively, this quantity is also equal to w divided by the expected distance between two selected consecutive positions — that we denote here by μ. Indeed, if we select a position every μ bases, we expect to choose w/μ positions on average in a window. Hence, with $dw \approx w/\mu$, we expect $d \approx \frac{1}{\mu}$. This intuitive link has already been noticed in the literature [5,30,17], but never, to the best of our knowledge, explicitly formalized as such, nor proved for any scheme. The purpose of this section is to formally establish this observation as a theorem, namely Theorem 1, under minimal assumptions.

The sequence S is represented by a vector of integers $\mathbf{R} = R_1, R_2, \dots,$ where $R_i = \texttt{rank}_{\mathcal{O}_m}(S[i : i+m]) \in [\![1, \sigma^m]\!]$ is the rank of the i-th m-mer of S, according to the order $\mathcal{O}_m$. A second integer vector, $\mathbf{P} = P_1, P_2, \dots,$ describes the positions of the selected minimizers when sliding a window of size w, i.e.

$$P_i = \min\{i \leq j \leq i + w - 1 : R_j = \min(R_i, \dots, R_{i+w-1})\}.$$

Note that $\mathbf{P}$ can still be defined no matter how minimizers are chosen, or more generally, a position, in a window. Since consecutive windows may choose the same position as a minimizer, consecutive values of P_i might be equal. It leads us to define the subsequence $\mathbf{P}^*$ of *distinct* values, i.e. $P_1^* = P_1$, P_2^* is the first P_i value distinct from P_1^*, P_3^* is the first P_i value distinct from P_1^* and P_2^*, and so on. In other words, $\mathbf{P}^*$ is the sequence corresponding to the set $\mathcal{S}_f(S)$ – i.e. the set of selected indices. Finally, we define $\mathbf{\Delta} = \Delta_1, \Delta_2, \dots$ as the sequence of distances between selected positions, i.e. $\Delta_1 = P_1^*$, and $\Delta_i = P_i^* - P_{i-1}^*$ for $i \geq 2$. An example is provided in Figure 1.

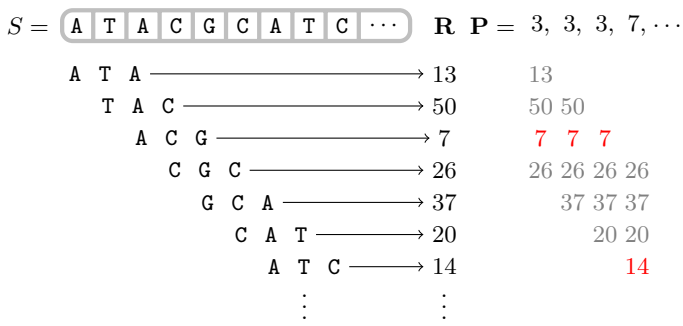


Fig. 1: An example of sequence S, represented as a sequence $\mathbf{R}$ of integers (here, using lexicographical order), and the sequence $\mathbf{P}$ of minimizers positions, with $m = 3$ and $w = 4$. We have $\mathbf{P}^* = 3, 7, \dots$ and $\mathbf{\Delta} = 3, 4, \dots$.

We put ourselves in the context where S is a random sequence, so that $\mathbf{R}, \mathbf{P}, \mathbf{P}^*$ and $\mathbf{\Delta}$ can be considered as vectors of random variables. Our goal is to establish a link between d and the expected distance between selected positions, i.e., between d and the $\mathbb{E}[\Delta_i]$'s. We obtain the following result.

Theorem 1. *Let $\mu > 0$. Consider $\varepsilon_i = \mathbb{E}[\Delta_i] - \mu$ as some realization of an underlying random variable ε with $\mathbb{E}[\varepsilon] = 0$. Then, we have $d \cdot \mu = 1$.*

Proof. The proof is deferred to Appendix S1.

Basically, Theorem 1 states that if we assume that the distances between consecutive selected positions of minimizers are somehow equally distributed, then their expected value is exactly $1/d$. This assumption is the minimum viable hypothesis for establishing the result (since μ must be well defined), and note also that this hypothesis does not depend on the process by which the R_i's values are obtained, nor how the positions of the minimizers $P_1, P_2, \ldots$ are selected in successive windows — thus making our result very general, applicable to *any local scheme*, and, to the best of our knowledge, the first of its kind.

At this point, we have not made any assumptions on the random vectors $\mathbf{R}, \mathbf{P}$ and $\mathbf{P}^*$. It is not trivial to formally verify whether the assumption regarding the $\mathbb{E}[\Delta_i]$'s is verified or not, as the variables are most likely not to be independent nor identically distributed. However, in practice, we can verify that the assumption holds. In the remainder of this section, we investigate how the assumption from Theorem 1 concerning the Δ_i's holds for random minimizers. The standard hypothesis that has been made in the literature regarding their choice is the following:

Hypothesis 1 ([5,6,35]) *Every m-mer in a window of length w has an equal probability of $1/w$ of being the smallest m-mer.*

As discussed in [35], while "not strictly true in practice", this hypothesis is reasonable and reflects reality accurately when using a randomized ordering". However, the formulation is unfortunate because it does not take into account the dependency between windows. If we consider a window $(R_i, \ldots, R_{i+w-1})$ outside of its context, then the hypothesis can be reformulated as $P_i \sim \mathcal{U}(\llbracket 1, w \rrbracket)$. Unfortunately, since the same minimizer will likely be selected by several successive windows, P_i cannot be considered independent, nor even identically distributed (since $P_i = P_{i-1}$ with a non-zero probability). Here, we propose starting from a more elementary hypothesis concerning the R_i's and observing, empirically and theoretically, the behavior of the variables derived from them, $\mathbf{P}$, $\mathbf{P}^*$ and $\mathbf{\Delta}$, in order to study the difference between Hypothesis 1 and reality, as well as the hypothesis concerning the expectation of the Δ_i's. Our working assumption is the following.

Hypothesis 2 *The random variables $R_1, R_2, \ldots$ are i.i.d, uniformly distributed over $\llbracket 1, \sigma^m \rrbracket$.*

Remark 1. Assuming a perfect hash function when computing the random order, our assumption states that the probability of collision between R_i's is $1/\sigma^m$ — which is consistent with [30, Lemma S7], that establishes that two distinct m-mers are identical in a random sequence with probability $1/\sigma^m$.

It is not too difficult to establish the law of $P_1 = P_1^* = \Delta_1$ from Hypothesis 2:

Proposition 1. *Under Hypothesis 2, $\forall 1 \leq i \leq w$,*

$$\mathbb{P}(P_1 = i) = \frac{1}{w} + \left(\frac{w-i}{w-1} - \frac{1}{2}\right) \cdot \frac{1}{\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right).$$

It follows that $P_1 \overset{a.s.}{\to} \mathcal{U}(\llbracket 1, w \rrbracket)$ when $m \to \infty$.

Proof. The proof is deferred to Appendix S2.

In Appendix S3, we observe that the limit distribution is reached very quickly, for fairly small values of m ($m = 8$ in our case), which implies that for all values of m likely to be used in practice, Hypothesis 1 is verified for P_1, and Δ_1 also follows a uniform distribution, meaning that $\mathbb{E}[\Delta_1] = \frac{w+1}{2}$. If, indeed, all subsequent Δ_i's have the same expectation, then, using Theorem 1, we retrieve the well-known density of random minimizers, $d = \frac{2}{w+1}$ [5,6].

Unfortunately, formally establishing the law of further variables becomes rapidly intractable due to the dependencies between them. To complete the study of subsequent $\mathbb{E}[\Delta_i]$'s, we resort to Monte-Carlo simulations of our probabilistic model and obtain the results depicted in Appendix S4. From our results, the assumption that $\mathbb{E}[\varepsilon] = 0$ is empirically verified (numerically, we get $\mathbb{E}[\varepsilon] \approx 0.0060$), and the $\mathbb{E}[\Delta_i]$'s are indeed distributed around $\mu = \frac{w+1}{2} = 5.5$ (using $w = 10$) — see supplementary Figure 8a. We obtain numerically a density of 0.1808, i.e. 0.55% of error compared to the theoretical value of $0.\overline{18}$. This empirical result highlights that Hypothesis 2 is indeed a proper assumption to work with, and also that the assumption of Theorem 1 is reasonable, since it is verified in practice.

2.3 Multiminimizers: trading time for space

Considering Theorem 1, obtaining a low density scheme on a sequence S requires having a scheme that selects positions as far as possible from one another in S, while still covering all k-mers from S. In this regard, the best possible minimizer scheme would select m-mers at a distance of $k - m + 1$ bases from each other. However, without a way to retrieve which m-mer was chosen for a specific k-mer, querying a k-mer would require checking if any of its $k - m + 1$ m-mers was selected as its minimizer. The same problem arises when inserting a k-mer into a database: one must perform $k - m + 1$ checks to verify whether it is already present or not.

To reduce this prohibitive cost, we propose a simple yet efficient solution leveraging the notion of super-k-mers. A *super-k-mer* is defined as a maximal sequence of consecutive k-mers having the same minimizer [36]. Our technique starts by generating N distinct hash functions (e.g. using N distinct random seeds). Then, we use these N hash functions to generate N distinct random minimizer schemes. Each of these minimizer schemes yields a set of super-k-mers on the sequence S, each covering all k-mers of S. When iterating over the minimizers of a sequence, among the super-k-mers that cover the first k-mer, we select the one that ends the farthest in the sequence. Then, we repeat this selection at the end of each selected super-k-mer: among all the super-k-mers covering the first uncovered k-mer, we select the one that ends the farthest in the sequence (see Figure 2). We detail our method in Algorithm 1, running in $O(N \cdot |S|)$ time and using $O(N \cdot w)$ memory.

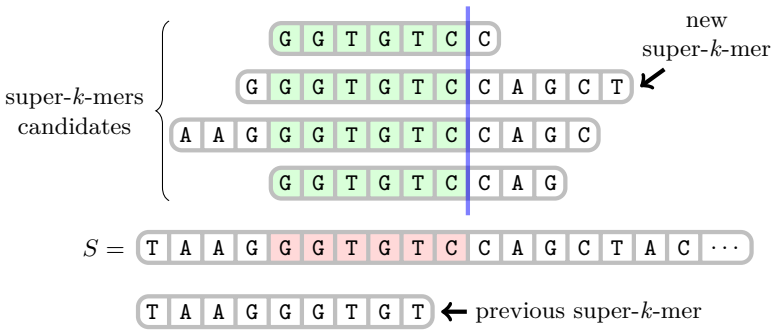


Fig. 2: Choice of a super-k-mer for $k = 6$ using a multiminimizer scheme. The first k-mer uncovered by the previous super-k-mer is highlighted in red. Among the $N = 4$ candidates super-k-mers covering this k-mer, the multiminimizer scheme selects the one ending the farthest in the sequence (i.e. farthest right of the blue line).

As mentioned in the introduction, a *local scheme* selects a m-mer in a k-mer solely based on the k-mer itself. This ensures covering all k-mers, or equivalently, all windows of $w = k - m + 1$ m-mers [33]. Formally, a local scheme is a function $f : \Sigma^k \to [\![1, w]\!]$. Importantly, the local scheme is both used at construction (e.g. when partitioning k-mers based on their minimizer) and query (e.g. to determine in which bucket a k-mer might be present). For a given k-mer $x \in \Sigma^k$, the position $f(x)$ is uniquely determined. While minimizers are local schemes, multiminimizers are not, as they differ drastically in their approach:

- At construction, we need to know when the previous super-k-mer ends, and where the candidates super-k-mers end, informations that a k-mer alone cannot provide. In this regard, the multiminimizer scheme has both to "remember the past" to determine when to select a new minimizer, and "delve into the future" to choose which minimizer candidate to keep.
- At query, for a given k-mer, we have to compute and lookup the N minimizers candidates to determine whether it has already been seen, since the broader context(s) in which it might have been seen are not known.

In the end, neither at construction nor query does the multiminimizer scheme behave like a local scheme. It is advantageous as it implies that its density is not subject to the lower bound of local schemes by Kille and Groot Koerkamp et al. [27] — although the $1/w$ bound obviously remains. As showed in upcoming Section 3.1, the multiminizer scheme does indeed beat this lower bound, and approach density of $1/w$ as N increases. This gain in density comes at the price of (bounded) increased query time, as per the discussion above.

Finally, note that the multiminimizer trick is not limited to random minimizer schemes, but could be used with any combination of N distinct local schemes (whether by varying parameters or using concurrent schemes) — in that sense, multiminimizers are a sort of *meta* scheme. To illustrate this point,

Algorithm 1: MultiMinimizers

```
Input: text T; window size w; minimizer size m; N minimizer schemes S_0, ..., S_{N-1}
Initialize buf of size N × w and run of size N
k ← w + m − 1
i* ← −w // starting position of the current super-k-mer
j* ← 0 // i* mod w
h* ← 0 // selected minimizer scheme
run[h*] ← 0 // selected minimizer scheme
for 0 ≤ i < |T| − k + 1 do
    if run[h*] = 0 then                    // the minimizer of the selected scheme has changed.
        j ← i mod w
        r ← i − i* // number of outdated values in the buffer
        for 0 ≤ h < N do
            for i* + w ≤ p < min(i* + w + r, |T| − k + 1) do
                buf[h][p mod w] ← S_h(T[p : p + k])
            run[h] ← 1
            while run[h] < w and buf[h][j + run[h] mod w] = buf[h][j] do
                run[h] ← run[h] + 1
        i*, j* ← i, j
        h* ← arg max run
    run[h*] ← run[h*] − 1
    yield buf[h*][j*]
```

in upcoming Section 3, we investigate two implementations of the multiminimizer trick,one with random minimizers, and the other with open-closed mod-minimizers [26].

2.4 On deduplicated density

It appears the standard density possesses a "twin" concept, that we call *deduplicated density* (which we denote by d^*), and that we define as the fraction of distinct minimizers needed to cover all k-mers of a set of sequences (or, equivalently, a set of k-mers), instead of the fraction of positions selected along a sequence. To the best of our knowledge, the deduplicated density has never been mentioned in the literature, even if it is implicitly linked to filter problems, such as the ones tackled by Needle [37].

Definition 5 (Minimizer Cover). *Let $\mathcal{X} = \{x_1, x_2, \dots\}$ be a set of (distinct) k-mers. The* minimizer cover *of $\mathcal{X}$, according to some minimizer scheme f, that we denote by $\textsc{MinCover}_f(\mathcal{X})$, is defined as $\textsc{MinCover}_f(\mathcal{X}) = \{\min_f(x) : x \in \mathcal{X}\}$, where $\min_f(x)$ designates the minimizer of x according to f.*

Analogously to standard density — recall Definition 4, we can define a *particular* deduplicated density $d^*(\mathcal{X})$ and obtain simple bounds, mimicking the ones we already have on standard particular density:

$$\frac{1}{|\mathcal{X}|} \leq d^*(\mathcal{X}) = \frac{|\textsc{MinCover}_f(\mathcal{X})|}{|\mathcal{X}|} \leq 1.$$

Standard density is computed as the limit of the expected particular density $\mathbb{E}[d(S)]$ on a random sequence S when $|S| \to \infty$, for fixed values of k and m — recall (1). However, one cannot have an arbitrarily large set $\mathcal{X}$ since $|\mathcal{X}| \leq \sigma^k$; therefore, we propose to define the deduplicated density simply as the expected particular density of a random set of k-mers $\mathcal{X}$, without any notion of limit.

Definition 6. *The (expected) deduplicated density of a minimizer scheme f is formally defined as*

$$d^* = \mathbb{E}_{\mathcal{X}}[d^*(\mathcal{X})] = \frac{1}{2^{\sigma^k} - 1} \sum_{\mathcal{X} \in \mathcal{P}(\Sigma^k) \setminus \emptyset} d^*(\mathcal{X}) \tag{2}$$

where $\mathcal{P}(\Sigma^k)$ designates the powerset of Σ^k.

Noticing that $1/|\mathcal{X}| \geq 1/\sigma^k$, we obtain the following trivial bounds for d^*.

Lemma 1. *For any minimizer scheme, we have $1/\sigma^k \leq d^* \leq 1$.*

Then, we have the following result.

Proposition 2. *Under Hypothesis 2, the deduplicated density of the random minimizer scheme is given by the following, where $w = k - m + 1$:*

$$d^* = \frac{1}{2^{\sigma^k} - 1} \sum_{n=1}^{\sigma^k} \binom{\sigma^k}{n} \frac{\sigma^m}{n} \left[1 - \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left[1 - \left(\frac{r}{\sigma^m} \right)^w + \left(\frac{r-1}{\sigma^m} \right)^w \right]^n \right].$$

Proof. The proof is deferred to Appendix S5.

The dependence on σ^k is unfortunate, making it intractable to compute d^* when k is large. We nonetheless computed the first values, shown in Figure 3 (left); one can see that d^* seems to be driven by the value of w rather than k, hence we conjecture that d^* might only depend on w. Besides, it seems that $d^* < d$ holds. To access higher values, we resorted to Monte-Carlo simulations, following the same probabilistic model as before, using Hypothesis 2. We observed how the standard density d and the deduplicated density d^* evolved over the course of parsing a random sequence, as shown in Figure 3 (right). While both densities seem to be equal for the first few thousand windows, they rapidly diverge. We looked into the data and observed that the number of distinct k-mers and distinct windows observed are equal (it is not surprising since the probability of collision of k-mers is negligible), so the difference between the densities is entirely explained by the discrepancy between the number of selected positions and the number of distinct minimizers. While those values are equal at first, soon we obtain repeated minimizers[3], hence the drop in d^*.

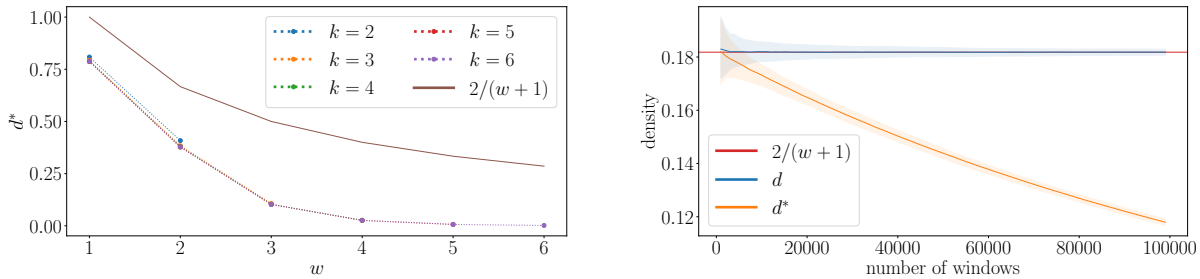


Fig. 3: (left) Theoretical values of d^* for $2 \leq k \leq 6$ as a function of $1 \leq w \leq k$, obtained with the formula of Proposition 2. (right) Monte-Carlo simulations of random minimizers under Hypothesis 2. We generated $N_{\text{simu}} = 10^3$ random sequences $\mathbf{R}$ of size $|\mathbf{R}| = 10^5$, with $m = 8$ and $w = 10$. For $i \geq 1$, we recorded the current values of d and d^* estimated on the sequence every $i \cdot 10^3$ windows. 95% of values across all simulations are within the colored areas, the solid lines being the medians.

There are obviously many more aspects of deduplicated density to be studied and discovered, but they are beyond the scope of this article. To conclude this section, we propose to consider applying multiminimizers to deduplicated density.

Definition 7 (Multiminimizer Cover). *Let $\mathcal{O}_m^1, \ldots, \mathcal{O}_m^N$ be N orders on m-mers, and let $\mathcal{X} = \{x_1, x_2, \ldots\}$ be a set of (distinct) k-mers. A set $\mathcal{Y} = \{y_1, y_2, \ldots\}$ of m-mers is said to be a* multiminimizer cover *of $\mathcal{X}$ if and only if, for any $x \in \mathcal{X}$, there exists $y \in \mathcal{Y}$ and $1 \leq i \leq N$, so that $\min_i(x) = y$, where $\min_i(x)$ designates the smallest m-mer of x according to the order $\mathcal{O}_m^i$. In other words, for each k-mer $x \in \mathcal{X}$, at least one of its N minimizer candidates belongs to $\mathcal{Y}$.*

The natural extension of the deduplicated density in this context would be $d^* = \frac{|\mathcal{Y}|}{|\mathcal{X}|}$, with the same trivial bounds as before. Contrary to $\textsc{MinCover}_f(\mathcal{X})$, we have here a choice of which candidate to choose as a minimizer, and therefore a chance at minimizing d^* by making the appropriate choices. Therefore, this leads to considering the following problem.

[3] While the *positions* of minimizers are somehow uniformly distributed within a window, as seen in Section 2.2, it is not the case of theirs *values* — that are the ones that repeat and therefore lead to a decrease in d^*. Indeed, the values of the minimizers are distributed as $\min(R_1, \ldots, R_w)$ where R_i are i.i.d. uniform variables, and therefore are strongly biased towards small values.

Problem 1 (MultiMinCover). Provided a set $\mathcal{X}$ of k-mers and N orders on m-mers $\mathcal{O}_m^1, \ldots, \mathcal{O}_m^N$, find a set $\mathcal{Y}$ of m-mers that is a multiminimizer cover of $\mathcal{X}$ and of minimum cardinality $|\mathcal{Y}|$.

This problem is closely related to the one of MinimumHittingSet: provided an integer k, and a set $\mathcal{X}$ of k-mers, find a set $\mathcal{Y}$ of m-mers (called an hitting set) such that each k-mer of $\mathcal{X}$ contains at least one m-mer from $\mathcal{Y}$, and of minimum cardinality. This problem is known to be NP-hard [38,39]. Here the problem is slightly different, as we do not ask the elements of $\mathcal{Y}$ to simply appear in the k-mers of $\mathcal{X}$, but also to be their minimizers for at least one of the orders. This problem also proves to be difficult, as per the following result.

Theorem 2. *MultiMinCover is NP-complete (for $\sigma \geq 3$ and $N \geq 2$).*

Proof. The proof is deferred to Appendix S6.

Since minimizing $|\mathcal{Y}|$ is difficult, we must resort to heuristics. Furthermore, when parsing long sequences, it would be prohibitively expensive to keep track of all the k-mers encountered and their candidates minimizers before starting to build the multiminimizer cover. For this reason, we have developed a heuristic that makes decisions throughout the parsing of a sequence, based solely on the local context, that is presented in Section 3.3.

3 Results

In this section, we evaluate the performance of two implementations of the multiminimizer technique: an iterator over random-minimizers-based multiminimizers, and a proof of concept of multiminimizers using open-closed mod-minimizers [26]. Both are written in the Rust programming language, and freely available at `github.com/lrobidou/multiminimizers`. We propose results for canonical k-mers, but our implementation also supports non-canonical k-mers.

Density results are provided in Section 3.1, whereas the space usage is investigated in Section 3.2 and the conservation of sampled multiminimizers is studied in Appendix S8. Regarding running time, note that the multiminimizer scheme must compute all candidate minimizers, so iteration over multiminimizers is linear in the number of hash functions. We measured the time required to iterate over a random sequence and confirmed that our implementation scales linearly with the number of hash functions, while remaining very fast as shown in Appendix S9. Finally, we provide some results in Section 3.3 on multiminimizers applied to filtering approaches.

3.1 Density

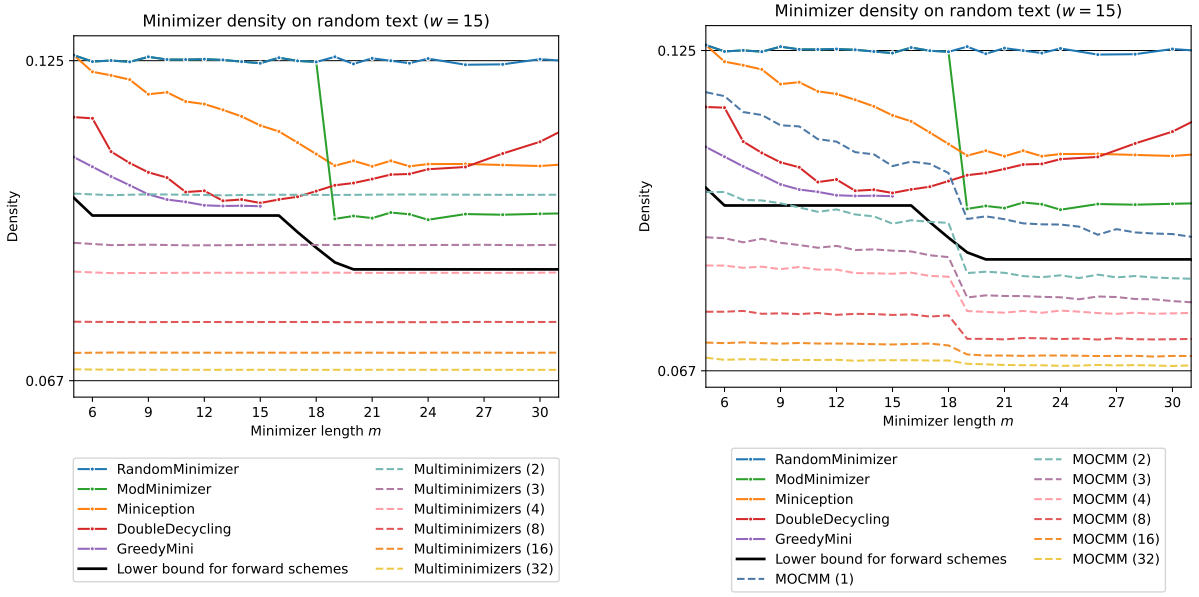
As discussed in Section 2.3, the multiminimizer trick aims at reducing the density of any local scheme. We expect to reduce the density when increasing the number of hash functions. The intuition is that increasing the number of super-k-mer candidates increases the chance of finding a super-k-mer that ends further than the others, thus increasing the distance between consecutive selected positions.

To show this density reduction, we generated a random sequence (5M bases) and computed the density of our scheme on it, for 1 to 8 hash functions, using $w = 15$ and varying m (thus k). We observed that the higher the number of hash functions, the lower the density, which allows us to reach densities lower than any other scheme (see Figure 4a). To our knowledge, multiminimizer is the first iterator over minimizers with a density lower than the lower bound introduced by [27]. Figures for additional m and w values are available in the Appendix S7.

Although multiminimizers are not a local scheme, we have observed that the empirical density $\hat{d}$ and the empirical average distance between selected positions $\hat{\mu}$ were indeed related as $\hat{d} \approx 1/\hat{\mu}$, hence showing that Theorem 1 might apply beyond the scope of local schemes.

To show that our multiminimizer trick can be applied to other schemes than random minimizers, we applied it to the open-closed mod-minimizer [26] to obtain a proof-of-concept of a multi open-closed mod-minimizer scheme (MOCMM for short). Figure 4b shows that this novel scheme achieves a lower density than the multiminimizers on a random sequence. At the time of writing, the library we used to compute the minimizer scheme does not support the open-closed mod-minimizer scheme. Therefore, we used a non-SIMD implementation, making it much slower. For this reason, we could only reasonably run it on a random sequence of 5×10^5 bases.

In both Figure 4a and Figure 4b, one can observe that the density approaches the theoretical lower bound $1/w$. This convergence seems intuitive since, when the number of hash functions becomes large enough, all m-mers of a k-mer are likely to be chosen as the minimizer candidate for at least one of the hash functions, and therefore the multiminimizer scheme tends to become optimal.



(a) Multiminimizers.

(b) Multi open-closed mod-minimizers.

Fig. 4: Density of multiple schemes, compared to our (a) multiminimizer and (b) MOCMM schemes with up to 32 hash functions. Our schemes are able to reach densities less than the lower bound of [27]. `GreedyMini` is close to this bound, but the computation time is exponential in m (the next data point would take $\approx$ 62h to compute).

3.2 Super-k-mers and hyper-k-mers space usage

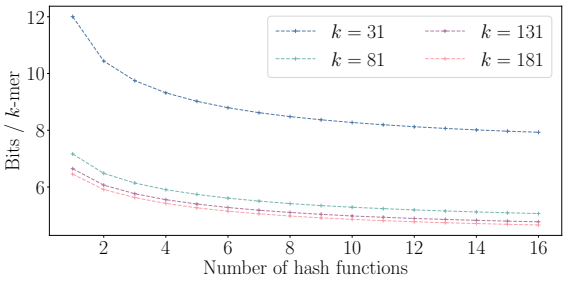
Since the space usage of a super-k-mer representation of a sequence is driven by the density of the underlying scheme [17, Thm. 1], we can translate a density reduction to a linear super-k-mer representation. Figure 5a shows the space usage of a super-k-mer representation of multiminizers, whereas Figure 5b provides the same information for MOCMMs, both for multiple numbers of hash functions and k values.

Ideally, to represent a sequence over an alphabet of size 4 (e.g., a DNA sequence S), one could use only 2 bits per nucleotide in S. However, super-k-mers use as low as 6 bits per nucleotide in S. Hyper-k-mers [17] were recently introduced and tend toward 4 bits per nucleotide in S when using a random-minimizer-based hyper-k-mer representation. This is still higher than the lower bound of 2 bits per nucleotide, due to random minimizers having a density of $\frac{2}{w+1}$.

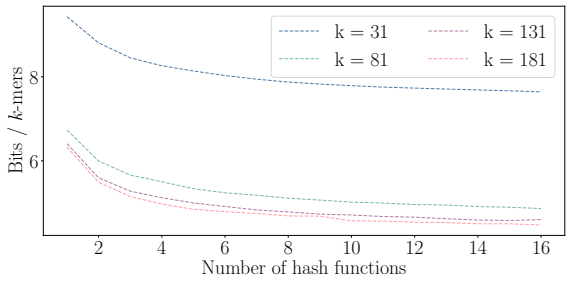
Since our multiminimizer scheme approaches a density of $\frac{1}{w}$, using a multiminimizer scheme could allow hyper-k-mers to tend toward 2 bits per nucleotide in the input sequence. To test this hypothesis, we replaced the iterator over minimizer of KFC (a hyper-k-mer-based k-mer counter introduced in [17]) by our iterator over multiminimizer and report the result in Figure 5c. Note that KFC is optimized for large k values (e.g. > 60), so we tested our hypothesis in this range. Our results show that KFC converges to 2 bits per base in S. To the best of our knowledge, this is the first streaming k-mer representation to achieve it.

3.3 Pin: a multiminimizer approach to Needle-like applications

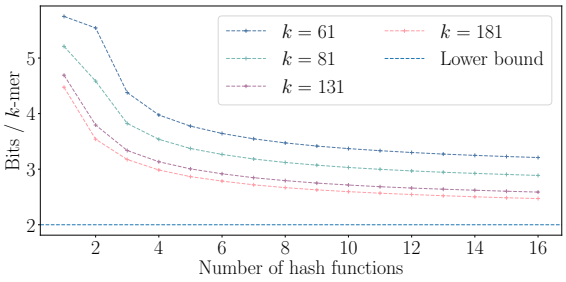
We developed a simple proof-of-concept minimizer index to demonstrate the utility of the multiminimizers approach for Needle-like filtering stages, in a prototype dubbed Pin (`github.com/Malfoy/Pin`). Such an index stores a set of minimizers that collectively cover all k-mers of interest, enabling efficient filtering as an indexed minimizer is a necessary condition for an indexed k-mer, and with sufficiently large minimizers, passing the filter already implies substantial sequence similarity. Our prototype relies on exact hash tables and handles a single uncolored set; the same idea can be readily adapted to colored or approximate index structures. During construction, we test the presence of k-mers and then greedily select minimizers to cover any uncovered sequences. We evaluate the approach on the Human HiFi accession `SRR11292123` (24Gb) in Figure 6, and observe a substantial reduction in index size with competitive build and query times. Merely switching from a single to two hash functions yields an index about 20% smaller, with roughly a 20% increase in construction time and about 85% in query time.



(a) Super-k-mer representation + multiminimizers.



(b) Super-k-mer representation + MOCMM.



(c) Hyper-k-mer representation + multiminimizers.

Fig. 5: Number of bits per k-mer for different representations of a random read, with $m = 21$, depending on k and the number of hash functions used with the multiminizer trick. Increasing the number of hash functions allows the hyper-k-mer representation (c) to converge to 2 bits/k-mer, the minimum required to represent a DNA sequence.

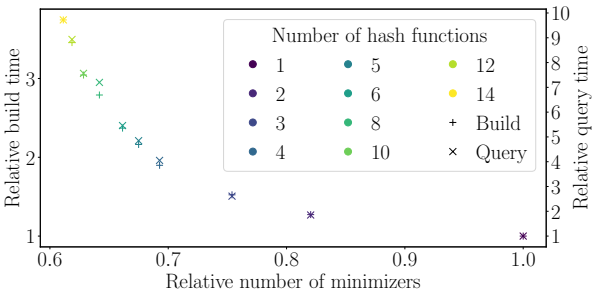


Fig. 6: Relative performances of the minimizer dictionary Pin using multiminimizers, comparing N hash functions vs $N = 1$. x-axis depicts the relative index size (in terms of the number of minimizers), and y-axis provides relative construction (left) and query (right) times. As an indicator, for $N = 1$, there are $\approx 850 \times 10^6$ minimizers, construction time is 234s, and query time is about 0.04s.

4 Future work

This work advances both the theory and practice of minimizer-based sampling by (i) establishing a general density–distance equivalence under minimal assumptions (**??**); (ii) introducing the novel notion of the deduplicated density, which captures the fraction of distinct minimizers covering a set of k-mers and for which we provide an analytical expression and empirical characterization in the random minimizers case; and (iii) designing the multiminimizer scheme, a simple yet effective scheme that trades time for space and achieves densities below the forward-scheme lower bound while preserving linear runtime. We also propose a proof-of-concept of a multi open-closed mod-minimizer scheme, achieving even lower densities. Empirically, increasing the number of hash functions consistently reduces density and memory footprint, and the benefits translate directly to super-k-mers, hyper-k-mers representations, and filtering indexes.

Several theoretical questions remain open. First of all, the theoretical density of the multiminimizer scheme remains to be studied. Since it empirically reaches densities lower than the bound for local schemes [27], it would be of great interest to find a way to express its density both in terms of the number and the underlying densities of the schemes used to build the multiminimizer meta scheme. Additionally, theoretical guarantees on the conservation of multiminimizers remain to be established. Then, the computational complexity of globally minimizing the standard density is still unknown, despite our NP-completeness result for the deduplicated counterpart; clarifying this complexity, including connections to polar-set formulations, is a natural next step. Beyond random minimizers, extending the analysis of deduplicated density to other scheme families and simplifying its expression in the random case would strengthen the theoretical picture. From an algorithmic standpoint, developing effective local or global heuristics for MultiMinCover is a promising path. Finally, on the practical side, two areas can be explored. Firstly, combining multiminimizers with other existing low-density schemes to form multi-seeded variants could further reduce density in practice while retaining fast iteration. The proof of concept "multiminimizers on open-closed mod-minimizer" could be vastly improved by, e.g., using SIMD instructions to speed up

computation. Secondly, replacing traditional minimizer iterators with our Rust library could improve a vast number of existing tools, beyond our work here on improving a hyper-k-mer-based k-mer counter. Thus, we set the stage for the next generation of theoretically grounded, robust, and efficient sampling schemes. Different sequence-mapping tools have highlighted the need for heuristics to fill in the deserts where sampled alignment landmarks are missing, as well as to better handle low complexity regions. We believe that the formalization of choice and selection-based schemes, such as those introduced in this work, will support improved alignment strategies in future developments.

Acknowledgments. This work was supported by the French National Research Agency AGATE [ANR-21-CE45-0012] and full-RNA [ANR-22-CE45-0007]. With financial support from ITMO Cancer of Aviesan within the framework of the 2021-2030 Cancer Control Strategy, on funds administered by Inserm. The authors are grateful to three anonymous reviewers for their valuable comments on a first version of the manuscript.

References

1. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology* **215**, 403–410 (1990).
2. Li, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**, 2103–2110 (2016).
3. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).
4. Marçais, G., Elder, C. & Kingsford, C. k-nonical space: sketching with reverse complements. *Bioinformatics* **40**, btae629 (2024).
5. Schleimer, S., Wilkerson, D. S. & Aiken, A. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 76–85 (2003).
6. Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M. & Yorke, J. A. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**, 3363–3369 (2004).
7. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).
8. Morisse, P., Marchet, C., Limasset, A., Lecroq, T. & Lefebvre, A. Consent: Scalable long read self-correction and assembly polishing with multiple sequence alignment. *BioRxiv* 546630 (2019).
9. Minkin, I. & Medvedev, P. Scalable pairwise whole-genome homology mapping of long genomes with bubbz. *IScience* **23** (2020).
10. Andreace, F., Pizzi, C. & Comin, M. Metaprob 2: metagenomic reads binning based on assembly using minimizers and k-mers statistics. *Journal of Computational Biology* **28**, 1052–1062 (2021).
11. Martayan, I. *et al.* Accelerating k-mer-based sequence filtering. *bioRxiv* 2025–06 (2025).
12. Constantinides, B., Lees, J. & Crook, D. W. Deacon: fast sequence filtering and contaminant depletion. *bioRxiv* 2025–06 (2025).
13. Ekim, B., Berger, B. & Chikhi, R. Minimizer-space de bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. *Cell systems* **12**, 958–968 (2021).
14. Ekim, B., Sahlin, K., Medvedev, P., Berger, B. & Chikhi, R. Efficient mapping of accurate long reads in minimizer space with mapquik. *Genome research* **33**, 1188–1197 (2023).
15. Deorowicz, S., Kokot, M., Grabowski, S. & Debudaj-Grabysz, A. KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics* **31**, 1569–1576 (2015). URL https://doi.org/10.1093/bioinformatics/btv022. https://academic.oup.com/bioinformatics/article-pdf/31/10/1569/49012901/bioinformatics_31_10_1569.pdf.
16. Kokot, M., Dugosz, M. & Deorowicz, S. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics* **33**, 2759–2761 (2017). URL https://doi.org/10.1093/bioinformatics/btx304. https://academic.oup.com/bioinformatics/article-pdf/33/17/2759/49040995/bioinformatics_33_17_2759.pdf.
17. Martayan, I., Robidou, L., Shibuya, Y. & Limasset, A. Hyper-k-mers: Efficient streaming k-mers representation. In Sankararaman, S. (ed.) *Research in Computational Molecular Biology*, 330–335 (Springer Nature Switzerland, Cham, 2025).
18. Chikhi, R., Limasset, A. & Medvedev, P. Compacting de bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* **32**, i201–i208 (2016).
19. Khan, J., Kokot, M., Deorowicz, S. & Patro, R. Scalable, ultra-fast, and low-memory construction of compacted de bruijn graphs with cuttlefish 2. *Genome biology* **23**, 190 (2022).
20. Cracco, A. & Tomescu, A. I. Extremely fast construction and querying of compacted and colored de bruijn graphs with ggcat. *Genome Research* **33**, 1198–1207 (2023).
21. Pibiri, G. E. Sparse and skew hashing of K-mers. *Bioinformatics* **38**, i185–i194 (2022). URL https://academic.oup.com/bioinformatics/article/38/Supplement_1/i185/6617506.
22. Marchet, C., Kerbiriou, M. & Limasset, A. Blight: efficient exact associative structure for k-mers. *Bioinformatics* **37**, 2858–2865 (2021). URL https://doi.org/10.1093/bioinformatics/btab217.
23. Zheng, H., Kingsford, C. & Marçais, G. Improved design and analysis of practical minimizers. *Bioinformatics* **36**, i119–i127 (2020).

24. Pellow, D. *et al.* Efficient minimizer orders for large values of k using minimum decycling sets. *Genome Research* **33**, 1154–1161 (2023).
25. Groot Koerkamp, R. & Pibiri, G. E. The mod-minimizer: A Simple and Efficient Sampling Algorithm for Long k-mers. In Pissis, S. P. & Sung, W.-K. (eds.) *24th International Workshop on Algorithms in Bioinformatics (WABI 2024)*, vol. 312 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 11:1–11:23 (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024). URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.WABI.2024.11.
26. Groot Koerkamp, R., Liu, D. & Pibiri, G. E. The open-closed mod-minimizer algorithm. *Algorithms for Molecular Biology* **20**, 4 (2025).
27. Kille, B., Groot Koerkamp, R., McAdams, D., Liu, A. & Treangen, T. J. A near-tight lower bound on the density of forward sampling schemes. *Bioinformatics* **41**, btae736 (2025).
28. Golan, S., Tziony, I., Kraus, M., Orenstein, Y. & Shur, A. Greedymini: generating low-density dna minimizers. *Bioinformatics* **41**, i275–i284 (2025).
29. Chikhi, R., Limasset, A., Jackman, S., Simpson, J. T. & Medvedev, P. On the representation of de bruijn graphs. In *International conference on Research in computational molecular biology*, 35–55 (Springer, 2014).
30. Zheng, H., Kingsford, C. & Marçais, G. Sequence-specific minimizers via polar sets. *Bioinformatics* **37**, i187–i195 (2021).
31. Hoang, M., Zheng, H. & Kingsford, C. Differentiable learning of sequence-specific minimizer schemes with deepminimizer. *Journal of Computational Biology* **29**, 1288–1304 (2022).
32. Pibiri, G. E. Sparse and skew hashing of k-mers. *Bioinformatics* **38**, i185–i194 (2022).
33. Marçais, G., DeBlasio, D. & Kingsford, C. Asymptotically optimal minimizers schemes. *Bioinformatics* **34**, i13–i22 (2018).
34. Pibiri, G. E., Shibuya, Y. & Limasset, A. Locality-preserving minimal perfect hashing of k-mers. *Bioinformatics* **39**, i534–i543 (2023).
35. Marçais, G. *et al.* Improving the performance of minimizers and winnowing schemes. *Bioinformatics* **33**, i110–i117 (2017).
36. Li, Y. *et al.* Memory efficient minimum substring partitioning. *Proceedings of the VLDB Endowment* **6**, 169–180 (2013).
37. Darvish, M., Seiler, E., Mehringer, S., Rahn, R. & Reinert, K. Needle: a fast and space-efficient prefilter for estimating the quantification of very large collections of expression experiments. *Bioinformatics* **38**, 4100–4108 (2022).
38. Orenstein, Y., Pellow, D., Marçais, G., Shamir, R. & Kingsford, C. Compact universal k-mer hitting sets. In *International Workshop on Algorithms in Bioinformatics*, 257–268 (Springer, 2016).
39. Orenstein, Y., Pellow, D., Marçais, G., Shamir, R. & Kingsford, C. Designing small universal k-mer hitting sets for improved analysis of high-throughput sequencing. *PLoS computational biology* **13**, e1005777 (2017).
40. Wald, A. On cumulative sums of random variables. *The Annals of Mathematical Statistics* **15**, 283–296 (1944).
41. Atkinson, K. E. *An introduction to numerical analysis* (John wiley & sons, 2008).

Supplementary material

S1 Proof of Theorem 1

We start by rewriting (1) as

$$d = \lim_{M\to\infty} \frac{1}{\sigma^M} \sum_{S\in\Sigma^M} \frac{|\mathcal{S}_f(S)|}{|S|-k+1} = \lim_{M\to\infty} \frac{1}{M-k+1} \sum_{S\in\Sigma^M} \frac{1}{\sigma^M}\cdot|\mathcal{S}_f(S)| = \lim_{M\to\infty} \frac{\mathbb{E}[|\mathcal{S}_f(S)|]}{M-k+1} \tag{3}$$

where $\mathbb{E}[|\mathcal{S}_f(S)|]$ designates the expected particular density of a (uniform) random sequence S of length M, seen as a random variable. Suppose we can extend the original sequence S, of size M, and continue selecting minimizers past the first M characters. We define $\tau_M = \inf\{n \in \mathbb{N} : P_n^* > M\}$ as the random variable that determines how many minimizers have been selected in S; hence $\tau_M = |\mathcal{S}_f(S)|$. Therefore, we can rewrite (3) as $d = \lim_{M\to\infty} \frac{\mathbb{E}[\tau_M]}{M-k+1}$.

Let us call $S_n = \sum_{i=1}^n \Delta_i$. Notice that, by definition of the Δ_i's, $S_n = P_n^*$; then $\tau_M = \inf\{n \in \mathbb{N} : S_n > M\}$ is a stopping time with respect to the σ-algebra generated by the Δ_i's. Using Wald's equation [40], we get

$$\mathbb{E}[S_{\tau_M}] = \mathbb{E}\left[\sum_{i=1}^{\tau_M} \mathbb{E}[\Delta_i]\right] = \mathbb{E}\left[\tau_M\cdot\mu + \sum_{i=1}^{\tau_M}\varepsilon_i\right] = \mathbb{E}[\tau_M]\cdot\mu + \mathbb{E}\left[\sum_{i=1}^{\tau_M}\varepsilon_i\right].$$

We apply again Wald's equation to the right-hand term and obtain

$$\mathbb{E}[S_{\tau_M}] = \mathbb{E}[\tau_M]\cdot\mu + \mathbb{E}\left[\sum_{i=1}^{\tau_M}\mathbb{E}[\varepsilon_i]\right] = \mathbb{E}[\tau_M]\cdot\mu + \underbrace{\mathbb{E}[\tau_M]\cdot\mathbb{E}[\varepsilon]}_{=0}.$$

By definition of S_n, τ_M and the Δ_i's, notice that $M \le S_{\tau_M} \le M + w - 1$. Therefore

$$\frac{M}{M-k+1} \le \frac{\mathbb{E}[\tau_M]}{M-k+1}\cdot\mu \le \frac{M+w-1}{M-k+1},$$

and, taking the limit as $M \to \infty$, we establish our result.

S2 Proof of Proposition 1

We begin with the following lemma.

Lemma 2. *For any function $f : [0,1] \to \mathbb{R}$ at least two times continuously derivable, we have*

$$\frac{1}{\sigma^m}\sum_{r=1}^{\sigma^m-1} f\left(\frac{r}{\sigma^m}\right) = \int_0^1 f(x)dx - \frac{f(1)+f(0)}{2\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right).$$

Proof. This is a direct application of the trapezoid rule for numerical integration. Defining $E = \int_0^1 f(x)dx - \frac{1}{\sigma^m}\left(\frac{f(1)+f(0)}{2} + \sum_{r=1}^{\sigma^m-1} f\left(\frac{r}{\sigma^m}\right)\right)$, we have $E = -\frac{1}{12\sigma^{2m}} f''(\eta)$ for some $\eta \in [0,1]$ [41, eq. (5.1.17)].

Let $1 \le i \le w$. For R_i to be the minimizer of the window $R_1, \dots, R_w$, then we must have the following situation :

$$\overbrace{R_1, \dots, R_{i-1}}^{>R_i}, \boxed{R_i}, \overbrace{R_{i+1}, \dots, R_w}^{\ge R_i}$$

The events $(R_j > R_i)$ and $(R_j \ge R_i)$ are clearly not independent of R_i; but resorting to the law of total probability, and that the R_j's are i.i.d., we obtain

$$\begin{aligned}
\mathbb{P}(P_1 = i) &= \sum_{r=1}^{\sigma^m} \mathbb{P}(R_1 > r)^{i-1}\cdot\mathbb{P}(R_1 \ge r)^{w-i}\cdot\mathbb{P}(R_i = r) \\
&= \frac{1}{\sigma^m}\sum_{r=1}^{\sigma^m}\left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1}\cdot\left(\frac{\sigma^m - r}{\sigma^m} + \frac{1}{\sigma^m}\right)^{w-i} \\
&= \frac{1}{\sigma^m}\sum_{r=0}^{\sigma^m-1}\left(\frac{r}{\sigma^m}\right)^{i-1}\cdot\left(\frac{r}{\sigma^m} + \frac{1}{\sigma^m}\right)^{w-i} \quad \text{by change of variable} \qquad (\star)
\end{aligned}$$

Using Lemma 2, with $f : x \mapsto x^{i-1}(x+\sigma^{-m})^{w-i}$, we obtain

$$\mathbb{P}(P_1 = i) = \frac{\mathbb{1}_{i=1}}{\sigma^{mw}} + \int_0^1 x^{i-1}(x+\sigma^{-m})^{w-i}dx - \frac{\left(1+\frac{1}{\sigma^m}\right)^{w-i} + \frac{\mathbb{1}_{i=1}}{\sigma^{mw}}}{2\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right).$$

Using the fact that $(x+\sigma^{-m})^n = x^n + \frac{nx^{n-1}}{\sigma^m} + O(\sigma^{-2m})$, we get

$$\begin{aligned}\mathbb{P}(P_1 = i) &= \int_0^1 x^{w-1}dx + \frac{w-i}{\sigma^m}\int_0^1 x^{w-2}dx - \frac{1}{2\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right)\\ &= \frac{1}{w} + \left(\frac{w-i}{w-1} - \frac{1}{2}\right)\cdot\frac{1}{\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right).\end{aligned}$$

S3 Convergence of P_1 to the uniform distribution

In Figure 7, we computed the distribution of P_1 for various values of m, using the exact value stated in ($\star$) rather than the approximation of Proposition 1. As one can see, P_1 converges quickly to the limit uniform distribution (as one can expect since the dependency on m is of the order of σ^{-m}).

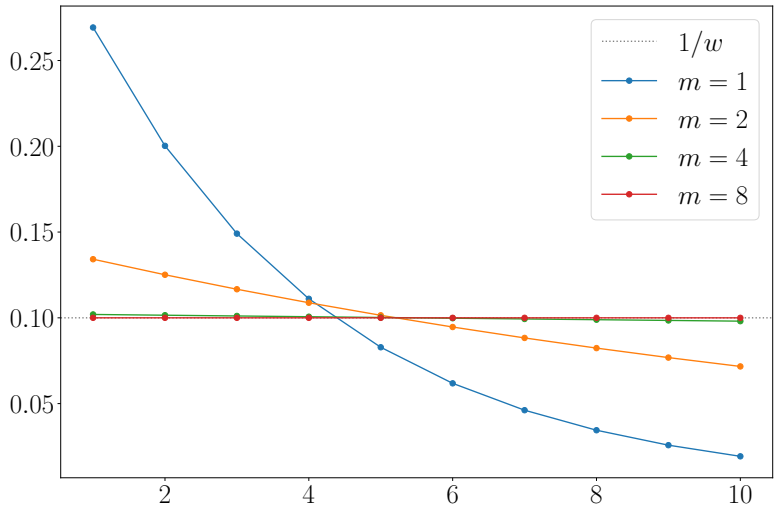
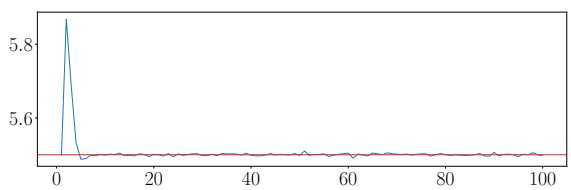


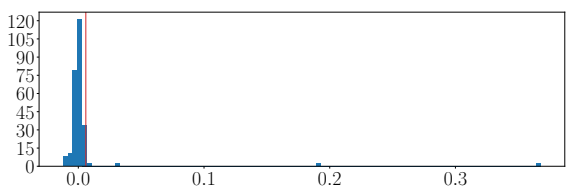
Fig. 7: With $w = 10$, evolution of the distribution of P_1 as m increases. The actual value of P_1 is computed, using ($\star$) instead of the approximations of Proposition 1.

S4 Monte-Carlo simulations of random minimizers

Simulating our probabilistic model of random minimizers, with Hypothesis 2, we obtain the results depicted in Figure 8, which have been commented on in the core of the paper.



(a) Empirical values of $\mathbb{E}[\Delta_i]$ for $1 \leq i \leq 100$. The red line is equal to $\mu = \frac{w+1}{2}$.



(b) Empirical distribution of ε. The red line denotes the empirical mean of ε.

Fig. 8: Monte-Carlo simulations of our model of random minimizers under Hypothesis 2. We used $m = 8$, $w = 10$ and generated $N_{\text{simu}} = 10^6$ random sequences $\mathbf{R}$, all long enough so that the sequence $\mathbf{\Delta} = \Delta_1, \Delta_2, \ldots$ contains at least 100 values.

One thing that seems surprising at first glance is the peak observed for $\mathbb{E}[\Delta_2]$. When sliding the window by one position, either the previous minimizer X is still in the window, and we change the

minimizer only if the new value X' is $< X$, or either X is not in the window anymore, and we select as a new minimizer the minimum value in the window (which is called a *rescan*). Here, P_1 is always obtained by a rescan operation, meaning that P_2^* is always obtained as "the next minimizer following a rescan". We do not observe any other peak in the distribution of $\mathbb{E}[\Delta_i]$'s as the other rescans are randomly found in the sequences, and therefore their impact is smoothed among all simulations. We have the following result; note that both the formula and the empirical simulations yield a value of $\mathbb{E}[\Delta_2] \approx 5.87$ for $w = 10$ and $m = 8$.

Proposition 3. *Assuming* $w^2 = o(\sigma^m)$, $\mathbb{E}[\Delta_2] = \dfrac{3\ln 2 - 1}{2} \cdot w + \dfrac{\ln 2}{2} + \dfrac{1}{8} - \dfrac{w-1}{32w^2} + O\left(\dfrac{1}{\sigma^m}\right)$.

We actually prove the following, more general, result:

Proposition 4.

$$\mathbb{E}[\Delta_2] = (H_{2w} - H_w)\frac{3w+1}{2} - \frac{w-1}{2} + O\left(\frac{1}{\sigma^m}\right)$$

where H_n *denotes the* n*-th harmonic number* $1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

Proof. To prove this, we need to distinguish two cases: either P_2^* "overwrites" P_1^* because $R_{P_2^*} < R_{P_1^*}$ or P_1^* goes out of scope and P_2^* is selected by a rescan. Therefore, we have

$$\mathbb{E}[\Delta_2] = \mathbb{E}[\Delta_2 \cdot \mathbb{1}_{P_2^* \text{ overwrites } P_1^*}] + \mathbb{E}[\Delta_2 \cdot \mathbb{1}_{\text{rescan happens}}]$$

In the following paragraphs, we show that

- $\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{P_2^* \text{ overwrites } P_1^*}] = w(H_{2w} - H_w) - \dfrac{w-1}{2} + O\left(\dfrac{1}{\sigma^m}\right)$.
- $\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{\text{rescan happens}}] = \dfrac{w+1}{2}(H_{2w} - H_w) + O\left(\dfrac{1}{\sigma^m}\right)$

Finally,

$$\begin{aligned}\mathbb{E}[\Delta_2] &= (H_{2w} - H_w) \cdot w - \frac{w-1}{2} + (H_{2w} - H_w) \cdot \frac{w+1}{2} + O\left(\frac{1}{\sigma^m}\right) \\ &= (H_{2w} - H_w)\frac{3w+1}{2} - \frac{w-1}{2} + O\left(\frac{1}{\sigma^m}\right).\end{aligned}$$

By approximating H_n as $\ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + O\left(\frac{1}{n^4}\right)$, we get the desired result:

$$\mathbb{E}[\Delta_2] = \frac{3\ln 2 - 1}{2} \cdot w + \frac{\ln 2}{2} + \frac{1}{8} - \frac{w-1}{32w^2} + O\left(\frac{1}{\sigma^m}\right).$$

Overwrite case. We start with the following lemma.

Lemma 3. *For any* $1 \le j \le i \le w$*, the probability that* $P_2^* = w + j$ *overwrites* $P_1^* = i$ *is*

$$\mathbb{P}((P_1^* = i) \cap (P_2^* = w+j) \cap (R_{w+j} < R_i)) = \frac{1}{w+j-1} - \frac{1}{w+j} + O\left(\frac{1}{\sigma^m}\right)$$

Proof. Let $O_{i,w+j}$ denote the event "$(P_1^* = i) \cap (P_2^* = w+j) \cap (R_{w+j} < R_i)$",

$$\begin{aligned}\mathbb{P}(O_{i,w+j}) &= \sum_{r=1}^{\sigma^m} \mathbb{P}(R_1 > r)^{i-1} \cdot \mathbb{P}(R_i = r) \cdot \mathbb{P}(R_1 \ge r)^{w-i+j-1} \cdot \mathbb{P}(R_{w+j} < r) \\ &= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \mathbb{P}(R_1 > r)^{i-1} \cdot \mathbb{P}(R_1 \ge r)^{w-i+j-1} \cdot (1 - \mathbb{P}(R_{w+j} \ge r)) \\ &= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \left(\frac{\sigma^m - r + 1}{\sigma^m}\right)^{w-i+j-1} \left(1 - \frac{\sigma^m - r + 1}{\sigma^m}\right) \\ &= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\frac{r-1}{\sigma^m}\right)^{i-1} \left(\frac{r}{\sigma^m}\right)^{w-i+j-1} \left(1 - \frac{r}{\sigma^m}\right) \quad \text{by change of variable}\end{aligned}$$

Using Lemma 2 with $f : x \mapsto (x - \sigma^{-m})^{i-1} x^{w-i+j-1}(1-x)$, we obtain

$$\mathbb{P}(O_{i,w+j}) = \int_0^1 (x - \sigma^{-m})^{i-1} x^{w-i+j-1}(1-x)dx - \frac{\mathbb{1}_{w+j=i+1}(-\sigma^{-m})^{i-1}}{2\sigma^m} + O\left(\frac{1}{\sigma^{2m}}\right)$$

Using the expansion $(x - \sigma^{-m})^{i-1} = x^{i-1} + O(\sigma^{-m})$, we get

$$\begin{aligned}\mathbb{P}(O_{i,w+j}) &= \int_0^1 x^{w+j-2}(1-x)dx + O\left(\frac{1}{\sigma^m}\right)\\ &= \frac{1}{w+j-1} - \frac{1}{w+j} + O\left(\frac{1}{\sigma^m}\right),\end{aligned}$$

since $\int_0^1 x^n(1-x)dx = \frac{1}{n+1} - \frac{1}{n+2}$.

Assuming $w^2 = o(\sigma^m)$,

$$\begin{aligned}\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{P_2^* \text{ overwrites } P_1^*}] &= \sum_{i=1}^{w}\sum_{j=1}^{i}(w-i+j)\cdot \mathbb{P}(O_{i,w+j})\\ &= \sum_{i=1}^{w}\sum_{j=1}^{i}(w-i+j)\left[\frac{1}{w+j-1} - \frac{1}{w+j} + O\left(\frac{1}{\sigma^m}\right)\right]\\ &= \sum_{i=1}^{w}\left[\sum_{j=1}^{i}\frac{w-i+j}{w+j-1} - \sum_{j=1}^{i}\frac{w-i+j}{w+j}\right] + O\left(\frac{1}{\sigma^m}\right)\\ &= \sum_{i=1}^{w}\left[\sum_{j=1}^{i}\frac{w-i+j-1+1}{w+j-1} - \left(i - i\sum_{j=1}^{i}\frac{1}{w+j}\right)\right] + O\left(\frac{1}{\sigma^m}\right)\\ &= \sum_{i=1}^{w}\left[i - (i-1)\sum_{j=1}^{i}\frac{1}{w+j-1} - i + i\sum_{j=1}^{i}\frac{1}{w+j}\right] + O\left(\frac{1}{\sigma^m}\right)\\ &= \sum_{i=1}^{w}\left[\sum_{j=1}^{i}\frac{1}{w+j-1} + i\left(\frac{1}{w+i} - \frac{1}{w}\right)\right] + O\left(\frac{1}{\sigma^m}\right)\end{aligned}$$

Expanding into three different sums, we have

$$\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{P_2^* \text{ overwrites } P_1^*}] = \underbrace{\sum_{i=1}^{w}\sum_{j=1}^{i}\frac{1}{w+j-1}}_{(A)} + \underbrace{\sum_{i=1}^{w}\frac{i}{w+i}}_{(B)} - \underbrace{\frac{1}{w}\sum_{i=1}^{w} i}_{(C)} + O\left(\frac{1}{\sigma^m}\right).$$

(C) is obviously $\frac{w+1}{2}$. For (A), note that the term $\frac{1}{w}$ appear w times, $\frac{1}{w+1}$ $w-1$ times, and so on up to $\frac{1}{2w-1}$ that appear only once. Hence,

$$(A) = \sum_{i=0}^{w-1}\frac{w-i}{w+i} = 1 + \sum_{i=1}^{w}\frac{w-i}{w+i} = 1 + w(H_{2w} - H_w) - \underbrace{\sum_{i=1}^{w}\frac{i}{w+i}}_{(B)}.$$

We do not need to compute (B) as terms cancel each other out, and

$$\begin{aligned}\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{P_2^* \text{ overwrites } P_1^*}] &= 1 + w(H_{2w} - H_w) - (B) + (B) - \frac{w+1}{2} + O\left(\frac{1}{\sigma^m}\right)\\ &= w(H_{2w} - H_w) - \frac{w-1}{2} + O\left(\frac{1}{\sigma^m}\right).\end{aligned}$$

Rescan case. Let $1 \leq r \leq \sigma^m$ and let $\alpha(r) = \frac{\sigma^m - r + 1}{\sigma^m}$. We use the trapezoid rule used in Lemma 2 to derive an alternative formula:

$$\frac{1}{\sigma^m} \sum_{s=1}^{\sigma^m - r} f\left(\frac{s}{\sigma^m}\right) = \int_0^{\alpha(r)} f(x)dx + O\left(\frac{1}{\sigma^m}\right) \tag{4}$$

where $f : [0,1] \to \mathbb{R}$ is twice continuously derivable.

Then, we have the following lemma.

Lemma 4. *For any $1 \leq i, j \leq w$, the probability that $P_2^* = i + j$ as a rescan when $P_1^* = i$ is*

$$\mathbb{P}((P_1^* = i) \cap (P_2^* = i + j) \cap (rescan)) = \frac{1}{w(w+i)} + O\left(\frac{1}{\sigma^m}\right)$$

Proof. Let us denote by $R_{i,j}$ the event "$(P_1^* = i) \cap (P_2^* = i + j) \cap (\text{rescan})$" for any $1 \leq i, j \leq w$. When this event is realized, then all variables

- $R_1, \ldots, R_{i-1}$ are $> R_i$;
- $R_{i+1}, \ldots, R_{w+i}$ are $\geq R_i$ (including R_{i+j});
- $R_{i+1}, \ldots, R_{i+j-1}$ are $> R_j$;
- $R_{i+j+1}, \ldots, R_{w+i}$ are $\geq R_j$

Therefore,

$$\begin{aligned}
\mathbb{P}(R_{i,j}) &= \sum_{r=1}^{\sigma^m} \sum_{s=1}^{\sigma^m} \mathbb{P}(R_1 > r)^{i-1} \cdot \mathbb{P}(R_i = r) \cdot \mathbb{P}((R_1 \geq r) \cap (R_1 > s))^{j-1} \\
&\qquad \cdot \mathbb{P}((R_{i+j} = s) \cap (R_{i+j} \geq r)) \cdot \mathbb{P}((R_1 \geq r) \cap (R_1 \geq s))^{w-j} \\
&= \frac{1}{\sigma^{2m}} \sum_{r=1}^{\sigma^m} \mathbb{P}(R_1 > r)^{i-1} \sum_{s=r}^{\sigma^m} \mathbb{P}(R_1 > s)^{j-1} \cdot \mathbb{P}(R_1 \geq s)^{w-j} \\
&= \frac{1}{\sigma^{2m}} \sum_{r=1}^{\sigma^m} \left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \sum_{s=r}^{\sigma^m} \left(\frac{\sigma^m - s}{\sigma^m}\right)^{j-1} \left(\frac{\sigma^m - s + 1}{\sigma^m}\right)^{w-j} \\
&= \frac{1}{\sigma^{2m}} \sum_{r=1}^{\sigma^m} \left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \sum_{s=0}^{\sigma^m - r} \left(\frac{s}{\sigma^m}\right)^{j-1} \left(\frac{s+1}{\sigma^m}\right)^{w-j}
\end{aligned}$$

Notice that the term $s = 0$ in the inner sum equals $\mathbb{1}_{j=1}\sigma^{-m(w-1)}$ which is $O(\sigma^{-m})$ as we suppose that $w > 1$. Applying (4) to the function $f : x \mapsto x^{j-1}(x + \sigma^{-m})^{w-j}$, we get

$$\begin{aligned}
\mathbb{P}(R_{i,j}) &= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \left[\int_0^{\alpha(r)} f(x)dx + O\left(\frac{1}{\sigma^m}\right)\right] \\
&= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \int_0^{\alpha(r)} f(x)dx\right) + O\left(\frac{1}{\sigma^m}\right)
\end{aligned}$$

Using the expansion $f(x) = x^{w-1} + O(\sigma^{-m})$, we get

$$\begin{aligned}
\mathbb{P}(R_{i,j}) &= \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \int_0^{\alpha(r)} x^{w-1}dx\right) + O\left(\frac{1}{\sigma^m}\right) \\
&= \frac{1}{w} \cdot \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\frac{\sigma^m - r}{\sigma^m}\right)^{i-1} \cdot \left(\frac{\sigma^m - r + 1}{\sigma^m}\right)^{w} + O\left(\frac{1}{\sigma^m}\right) \\
&= \frac{1}{w} \cdot \frac{1}{\sigma^m} \sum_{r=1}^{\sigma^m} \left(\frac{r-1}{\sigma^m}\right)^{i-1} \cdot \left(\frac{r}{\sigma^m}\right)^{w} + O\left(\frac{1}{\sigma^m}\right)
\end{aligned}$$

Applying Lemma 2 with the function $f : x \mapsto (x - \sigma^{-m})^{i-1}x^w$, we get

$$\mathbb{P}(R_{i,j}) = \frac{1}{w} \int_0^1 (x - \sigma^{-m})^{i-1} x^w dx + O\left(\frac{1}{\sigma^m}\right) = \frac{1}{w(w+i)} + O\left(\frac{1}{\sigma^m}\right)$$

Finally, still assuming that $w^2 = o(\sigma^m)$

$$\begin{aligned}\mathbb{E}[\Delta_2 \cdot \mathbb{1}_{\text{rescan happens}}] &= \sum_{i=1}^{w}\sum_{j=1}^{w} j \cdot \mathbb{P}(R_{i,j}) = \sum_{i=1}^{w}\sum_{j=1}^{w} \frac{j}{w(w+i)} + O\left(\frac{1}{\sigma^m}\right) \\ &= \frac{w+1}{2}\sum_{i=1}^{w}\frac{1}{w+i} + O\left(\frac{1}{\sigma^m}\right) = \frac{w+1}{2}(H_{2w} - H_w) + O\left(\frac{1}{\sigma^m}\right),\end{aligned}$$

as claimed.

S5 Proof of Proposition 2

We are interested here in computing the deduplicated density d^* of the random minimizer scheme. We have, from (2),

$$d^* = \frac{1}{2^{\sigma^k}-1}\sum_{\mathcal{X}\in\mathcal{P}(\Sigma^k)\setminus\emptyset} d^*(\mathcal{X}) = \frac{1}{2^{\sigma^k}-1}\sum_{n=1}^{\sigma^k}\sum_{\substack{\mathcal{X}\in\mathcal{P}(\Sigma^k)\setminus\emptyset \\ |\mathcal{X}|=n}} d^*(\mathcal{X}) = \frac{1}{2^{\sigma^k}-1}\sum_{n=1}^{\sigma^k}\binom{\sigma^k}{n}\mathbb{E}_{|\mathcal{X}|=n}[d^*(\mathcal{X})]$$

To compute the required expectation, let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of k-mers, chosen uniformly at random. We denote by $M_i \in [\![1, \sigma^m]\!]$ the random variable denoting the minimizer of k-mer X_i (assimilating minimizers and their rank according to the chosen order $\mathcal{O}_m$). Under Hypothesis 2, it is not difficult to establish that $\mathbb{P}(M_i > r) = \mathbb{P}(X_1 > r)^w$ where X_i denotes the i-th m-mer of X_i and $w = k - m + 1$. Therefore, we obtain

$$\mathbb{P}(M_i = r) = \left(\frac{\sigma^m - r + 1}{\sigma^m}\right)^w - \left(\frac{\sigma^m - r}{\sigma^m}\right)^w.$$

We denote by P_r the number of k-mers in $\mathcal{X}$ that admits r as their minimizers; and we are interested in computing

$$\begin{aligned}\mathbb{E}[|\textsc{MinCover}(\mathcal{X})|] &= \sum_{r=1}^{\sigma^m}\mathbb{E}[\mathbb{1}_{P_r>0}] = \sum_{r=1}^{\sigma^m}(1 - \mathbb{P}(P_r = 0)) = \sigma^m - \sum_{r=1}^{\sigma^m}(1 - \mathbb{P}(M_1 = r))^n \\ &= \sigma^m\left[1 - \frac{1}{\sigma^m}\sum_{r=1}^{\sigma^m}\left[1 - \left(\frac{\sigma^m - r + 1}{\sigma^m}\right)^w + \left(\frac{\sigma^m - r}{\sigma^m}\right)^w\right]^n\right] \\ &= \sigma^m\left[1 - \frac{1}{\sigma^m}\sum_{r=1}^{\sigma^m}\left[1 - \left(\frac{r}{\sigma^m}\right)^w + \left(\frac{r-1}{\sigma^m}\right)^w\right]^n\right]\end{aligned}$$

This is exactly the standard coupon collector's problem, where the coupons are distinct minimizers and $|\mathcal{X}|$ is the number of coupons bought.

S6 Proof of Theorem 2

We prove the NP-completeness of MultiMinCover by (i) showing first that the problem is in NP and then (ii) by reduction from SetCover. One anonymous reviewer also suggested an alternative reduction from VertexCover, which is presented at the end of this section.

MultiMinCover is in NP. For this, we look at the decision version of MultiMinCover. Provided an integer k, a set $\mathcal{X}$ of k-mers and N orders on m-mers $\mathcal{O}_m^1, \ldots, \mathcal{O}_m^N$, is there a set $\mathcal{Y}$ of m-mers that is a multiminimizer cover of $\mathcal{X}$ and with cardinality $|\mathcal{Y}| \leq k$? If one provides a set $\mathcal{Y}$ of m-mers of cardinality $\leq k$, one can trivially check whether $\mathcal{Y}$ is a multiminimizer cover of $\mathcal{X}$ by simply iterating over k-mers in $\mathcal{X}$, compute their N minimizers and check whether at least one of them belong to $\mathcal{Y}$. This can be done in $O(|\mathcal{X}| \cdot N)$, hence the result.

Reduction from SetCover. Let $\mathcal{U} = \{1, \ldots, n\}$ and let $\mathcal{S} \subseteq 2^{\mathcal{U}}$ be a collection of subsets of $\mathcal{U}$, whose union is $\mathcal{U}$. We recall that the SetCover problem amounts to finding the smallest subcollection of $\mathcal{S}$ whose union is $\mathcal{U}$. The *frequency* of an element $u \in \mathcal{U}$ is defined at the number of subsets in $\mathcal{S}$ that contains u. We define $N = \max_{u\in\mathcal{U}} \text{freq}(u)$ as the maximum frequency of any element of $\mathcal{U}$. Note that $N \leq |\mathcal{S}|$.

Example 1. To illustrate the proof, we propose to follow an example. Let $\mathcal{U} = \{1, \ldots, 5\}$ and $\mathcal{S} = \{\{1,2,3\},\{4,5\},\{2,4\},\{3,4\}\}$. The optimal solution for this instance of set cover is $\{1,2,3\},\{4,5\}$ of cardinality 2. The maximal frequency is $N = 3$ (since 4 appears in 3 subsets).

We use the following 3-letter alphabet $\Sigma = \{0, 1, a\}$. Let $m = \lceil \log_2 |\mathcal{S}| \rceil$. Take any indexing $S_1, \ldots, S_{|\mathcal{S}|}$ of the subsets forming $\mathcal{S}$; we associate to each subset S_i the m-mer P_i corresponding to i in binary coded with m bits. Then, let $k = Nm+N-1 = N(m+1)-1$. To each element $u \in \mathcal{U}$, let $i_1 \ldots, i_f$ be the indices of the S_i's containing u, with $f = \mathrm{freq}(u)$. We associate to u the k-mer $X_u = m_{i_1} a m_{i_2} a \ldots a m_{i_f} a^{(N-f)(m+1)}$.

Example 2. We have $m = \lceil \log_2 4 \rceil = 2$. We associate $m_1 = 00$ to $\{1,2,3\}$, $m_2 = 01$ to $\{4,5\}$, $m_3 = 10$ to $\{2,4\}$ and $m_4 = 11$ to $\{3,4\}$. We have $k = 8$ and the following k-mers

$$X_1 = \underline{00}aaaaaa, \quad X_2 = \underline{00}a\underline{10}aaa, \quad X_3 = \underline{00}a\underline{11}aaa, \quad X_4 = \underline{01}a\underline{10}a\underline{11}, \quad X_5 = \underline{01}aaaaaa.$$

The goal now is to exhibit N orders $\mathcal{O}_m^1, \ldots, \mathcal{O}_m^N$ on m-mers, so that the following is true: for each k-mer x, its N minimizers correspond exactly to its m-mers coming from the m_i's; i.e. its minimizers contain only the letters 0 and 1. It should be clear that any solution to MultiMinCover in this context is equivalent to a solution of the original SetCover instance. Since there are $|\mathcal{U}| = n$ k-mers of size $N(m+1) - 1 = O(|\mathcal{S}| \log_2 |\mathcal{S}|)$ and exactly $|\mathcal{S}|$ minimizers to chose from, the reduction is indeed polynomial, hence the result — once we prove said orders on m-mers can indeed be constructed in polynomial time.

By construction, we want the m-mers $m_1, \ldots, m_{|\mathcal{S}|}$ to receive a rank (for all orders) between 1 and $|\mathcal{S}|$; all subsequent possible m-mers (either binary, or containing at least one 'a') should receive a rank $> |\mathcal{S}| + 1$, and their relative order is of no importance since they will not be selected as minimizers whatsoever. By doing so, we ensure that the N minimizers of each k-mer X_u can only be the m_i's it is made of. If X_u contains several such m_i's, the only thing left to do is to ensure that each of them is indeed a minimizer for at least one order – so that the number of distinct minimizers of X_u is equal to the frequency of u in the original SetCover instance — thus ensuring the 1-to-1 correspondence between solutions of the two problems.

We exhibit Algorithm 2 to construct those orders.

Example 3. At this point, we have determined that any m-mer containing at least one 'a' has rank at least 5 in any of the 3 orderings to build, and $\{00, 01, 10, 11\}$ must receive ranks 1 to 4. We initialize two arrays, `ranks` and `minimizers`, following Algorithm 2. Applying Lines 7-9 to X_1 and X_5, we get:

`rank`	$\mathcal{O}_m^1$	$\mathcal{O}_m^2$	$\mathcal{O}_m^3$
00			
01			
10			
11			

`minimizers`	$\min_1(x)$	$\min_2(x)$	$\min_3(x)$	`cand`(x)
X_1	00	00	00	$\emptyset$
X_2				00, 10
X_3				00, 11
X_4				01, 10, 11
X_5	01	01	01	$\emptyset$

Now, consider X_2 and the m-mer 00. Applying Lines 11-15, choosing $l = 1$, we get:

`rank`	$\mathcal{O}_m^1$	$\mathcal{O}_m^2$	$\mathcal{O}_m^3$
00	1		
01			
10			
11			

`minimizers`	$\min_1(x)$	$\min_2(x)$	$\min_3(x)$	`cand`(x)
X_1	00	00	00	$\emptyset$
X_2	00			10
X_3	00			11
X_4				01, 10, 11
X_5	01	01	01	$\emptyset$

For X_2 and X_3, their respective remaining candidate fill the empty spots in `minimizers` by applying again Lines 7-9. Considering X_4, a possible outcome could be the following:

`rank`	$\mathcal{O}_m^1$	$\mathcal{O}_m^2$	$\mathcal{O}_m^3$
00	1		
01	2		
10		1	
11			

`minimizers`	$\min_1(x)$	$\min_2(x)$	$\min_3(x)$	`cand`(x)
X_1	00	00	00	$\emptyset$
X_2	00	10	10	$\emptyset$
X_3	00	11	11	$\emptyset$
X_4	01	10	11	$\emptyset$
X_5	01	01	01	$\emptyset$

After which, any completion of `ranks` would work.

Algorithm 2: DETERMINEORDERS

Input: $m_1, \ldots, m_{|\mathcal{S}|}$; $X_1, \ldots, X_n$; N
1 Initialize `ranks`, a $|\mathcal{S}| \times N$ array
2 Initialize `minimizers`, a $n \times N$ array
3 **for** $1 \leq i \leq n$ **do**
4 Let $\texttt{cand}(X_i) \subseteq \{m_1, \ldots, m_{|\mathcal{S}|}\}$ be the set of candidates minimizers of X_i
5 **while** $\exists i, \mathit{cand}(X_i) \neq \emptyset$ **do**
6 Let $m_j \in \texttt{cand}(X_i)$
7 **if** $\mathit{cand}(X_i) \setminus \{m_j\} = \emptyset$ **then**
 `// The relative rank of` m_j `does not matter here, so we leave it be.`
8 **for** *all* l *so that* $\mathit{minimizers}(X_i, l)$ *is not defined* **do**
9 $\texttt{minimizers}(X_i, l) \leftarrow m_j$
10 **else**
11 Let l so that $\texttt{minimizers}(X_i, l)$ is not defined
12 Define $\texttt{ranks}(m_j, l)$ as the next integer between 1 and $|\mathcal{S}|$ not already used in column $\texttt{ranks}(\cdot, l)$
13 **for** *all* x' *so that* $m_j \in \mathit{cand}(x')$ **do**
14 **if** $\mathit{minimizers}(x', l)$ *is not defined* **then**
15 $\texttt{minimizers}(x', l) \leftarrow m_j$
16 Remove m_j from $\texttt{cand}(X_i)$
 `// At this point, the remaining ranks do not matter, so any completion will do.`
17 **for** $1 \leq l \leq N$ **do**
18 Attribute empty values $\texttt{ranks}(\cdot, l)$ at random among the remaining integers up to $|\mathcal{S}|$

Proposition 5. *Algorithm 2 is correct and runs in polynomial time.*

Proof. Each iteration in the `while` loop eliminates one candidate minimizer, of which there are at most $|\mathcal{U}| \cdot N$. For a given candidate minimizer, if Lines 7-9 are executed, the `for` loop takes at most N steps; and if Lines 11-15 are executed, parsing a column in `ranks` takes $|\mathcal{S}|$ steps, and the `for` loop at most $|\mathcal{U}|$ steps. Since $N \leq |\mathcal{S}|$, we end up with a worst-case complexity of $O\big(|\mathcal{U}| \cdot |\mathcal{S}| \cdot (|\mathcal{S}| + |\mathcal{U}|)\big)$, polynomial as claimed.

For the algorithm to be correct, we must never encounter the case where there exists X_i and $m_j \in \texttt{cand}(X_i)$ such that $\texttt{minimizers}(X_i, l)$ is already defined for all $1 \leq l \leq N$. In such a case, m_j could not be a minimizer for X_i. Suppose we are in such a case. Let $\texttt{cand}_I(X_i)$ be the set of candidates of X_i at the initialization of the algorithm. Since $m_j \in \texttt{cand}_I(X_i)$ and since $\texttt{minimizers}(X_i, l)$ is defined for all l, then surely there exists $m_{j'} \in \texttt{cand}_I(X_i)$, $j' \neq j$, that appears at least twice in the list of minimizers of X_i, by pigeonhole principle ($|\texttt{cand}_I(X_i) \setminus \{m_j\}| < N$). When a minimizer is affected as a consequence of Lines 11-15, it is removed from the list of candidates (and therefore cannot be added again as a minimizer for another order), so the only way $m_{j'}$ could have been affected to two orders if as a consequence of Lines 7-9 — but there are only triggered if $m_{j'}$ is the only remaining element of $\texttt{cand}(X_i)$, thus a contradiction since it still contain at least m_j.

Reduction from VERTEXCOVER. This alternative reduction was proposed by an anonymous reviewer. Let $G = (V, E)$ be graph. Using the same formalism as above, associate to each vertex i a distinct minimizer $m_i = \texttt{binary}(i)$ encoded with $m = \lceil |V| \rceil$ bits. As above, we use the alphabet $\Sigma = \{0, 1, a\}$.

For each edge $(i, j) \in E$, assume w.l.o.g $m_i < m_j$, we construct the k-mer $X_{(i,j)} = m_i a m_j$. We define two orders $\mathcal{O}^1$ and $\mathcal{O}^2$ as $(m_i, m_j) \in \mathcal{O}^1 \iff m_i < m_j$ and $(m_i, m_j) \in \mathcal{O}^2 \iff m_i > m_j$. Finally, we complete the orders by saying that $x < y$ if a appears in y and not in x, for both orders $\mathcal{O}^1$ and $\mathcal{O}^2$. If a belong to both x and y, any arbitrary order can do. In the end, for all k-mers $X_{(i,j)}$, $\min_1(X_{(i,j)}) = m_i$ and $\min_1(X_{(i,j)}) = m_j$. Let $\mathcal{X} = \{X_{(i,j)} : \{i, j\} \in E, m_i < m_j\}$.

Lemma 5. *VERTEX COVER admits a yes-instance (V, E, k) iff MULTIMINCOVER instance $(\mathcal{X}, \mathcal{O}^1, \mathcal{O}^2, k)$ is a yes-instance.*

Proof. ($\Rightarrow$) If (V, E, k) is a yes-instance with vertex cover $\mathcal{C}$, then the set $\mathcal{Y} = \{m_i : i \in \mathcal{C}\}$ is a multiminizer cover of size k for $\mathcal{X}$ since for any k-mer $X_{(i,j)}$ either $\min_1(X_{(i,j)}) = m_i$ or $\min_2(X_{(i,j)}) = m_j$.

(⇐) Let (V, E, k) be a no-instance. Assume that our constructed instance $(\mathcal{X}, \mathcal{O}^1, \mathcal{O}^2, k)$ is a yes-instance with multiminimizer cover $\mathcal{Y}$. Then every $X_{(i,j)} \in \mathcal{X}$ has a minimizer $\min_1(X_{(i,j)}) = m_i$ or $\min_2(X_{(i,j)}) = m_j$ per definition of $\mathcal{O}^1$ and $\mathcal{O}^2$; and the set $\mathcal{C} = \{i : m_i \in \mathcal{Y}\}$ is a vertex cover of size k for $G = (V, E)$ with $E = \{(i, j) : X_{(i,j)} \in \mathcal{X}\}$.

S7 Density plots for multiple w values

In this section, we ran the same evaluation as in Section 3.1, but applied our multiminimizer iterator to multiple w and more m values to check that our results are robust with regard to w and m. Figure 9 and Figure 10 shows that our results are indeed robust with regard to w: for all w values, our iterator can yield densities lower than the lower bound.

S8 Conservation with respect to error rate

In this section, we study the evolution of the conservation of the sampled multiminimizers with respect to the error rate and the number of hash functions. Given a sequence S and a mutated version of the sequence S' for a given error rate, we measure the conservation using the Jaccard similarity of $A = \textsc{MultiMinimizers}(S)$ and $B = \textsc{MultiMinimizers}(S')$, i.e. $\frac{|A \cap B|}{|A \cup B|}$. The results are shown in Figure 11. In particular, we can observe that the conservation decreases slightly faster with respect to the error rate as the number of hash functions increases.

S9 Linearity of indexation time

In this section, we show that the iteration over multiminimizers is linear with regard to the number of hash functions. The results are shown in Figure 12.

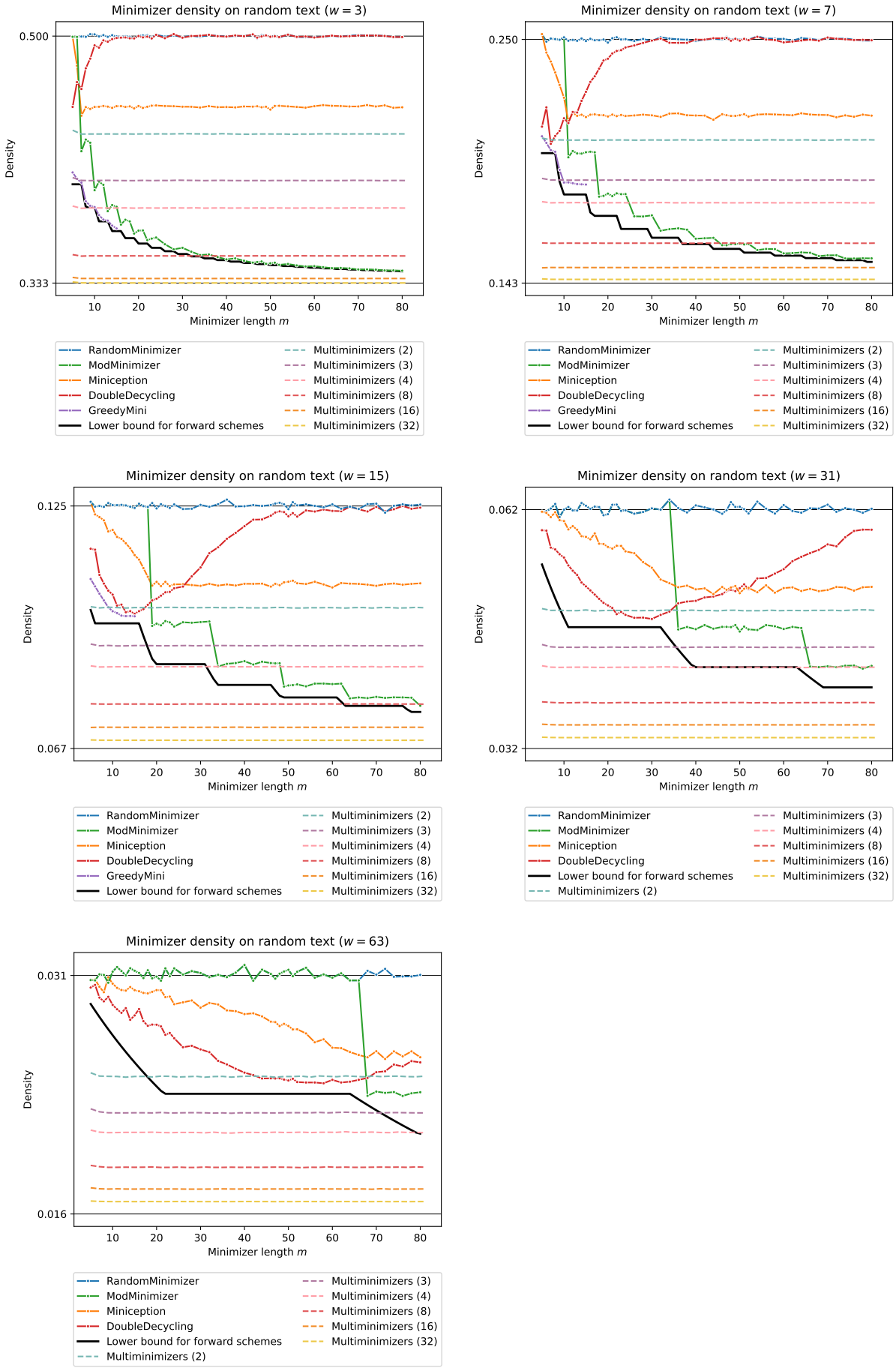


Fig. 9: Density of multiple schemes, including multiminimizers with up to 32 hash functions. Same plot as Figure 4a, for $w \in \{3, 7, 31, 63\}$.

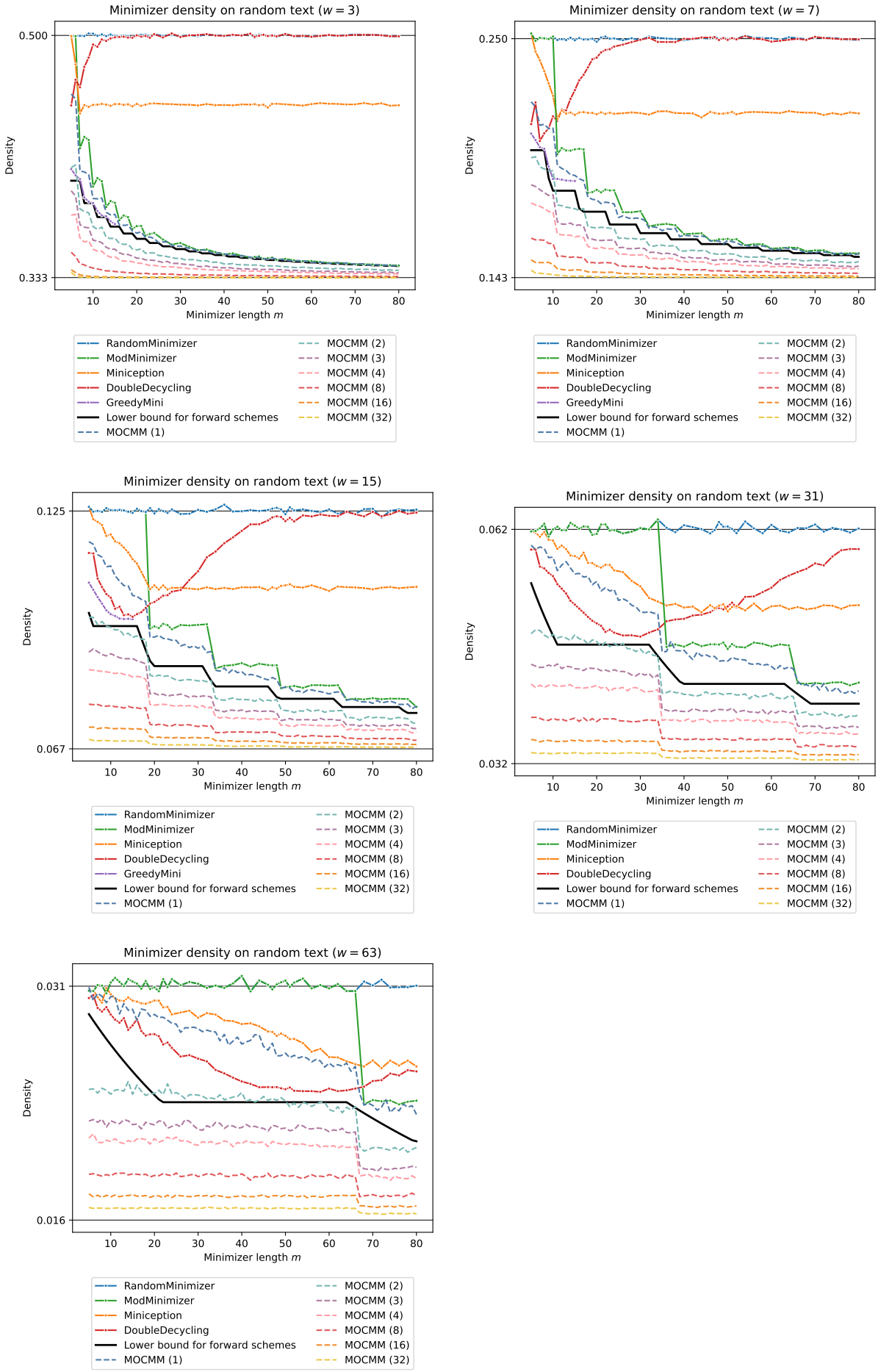


Fig. 10: Density of multiple schemes, including multiminimizers based on open-closed mod-minimizers [26] with up to 32 hash functions. Same plot as Figure 4b, for $w \in \{3, 7, 31, 63\}$.

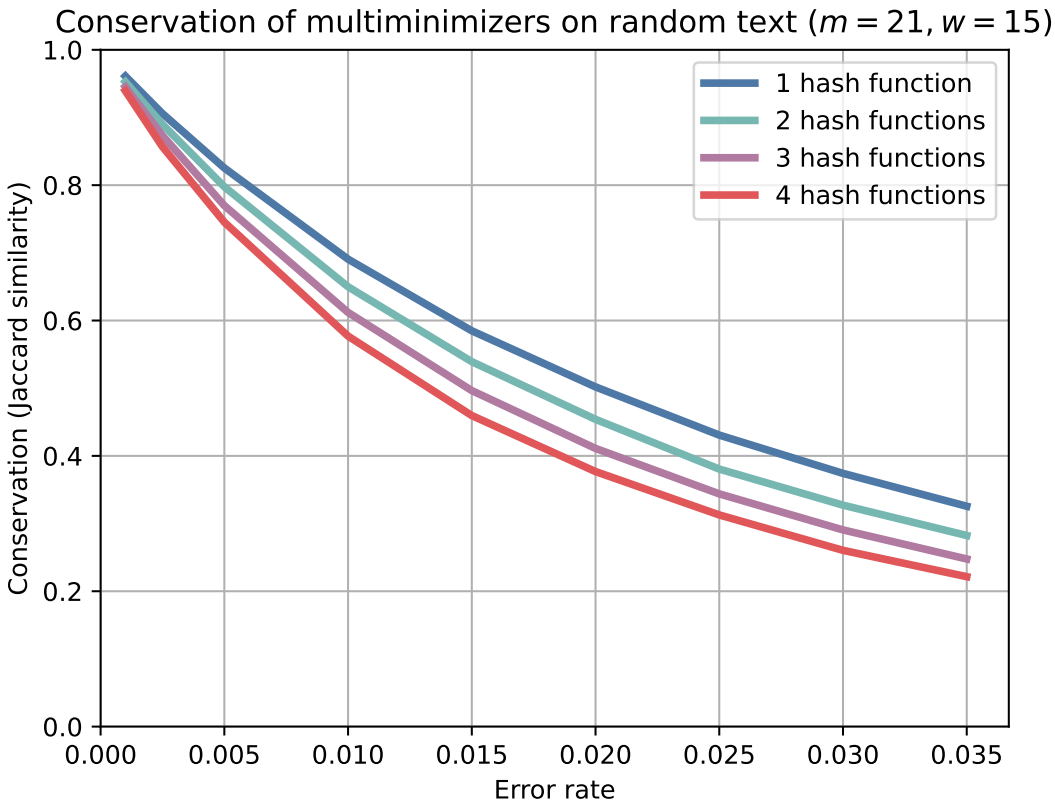


Fig. 11: Conservation of sampled multiminimizers, measured by their Jaccard similarity, with $1 \leq N \leq 4$ for a random string of size 10^5 compared to a mutated version of the string with up to 3.5% of errors.

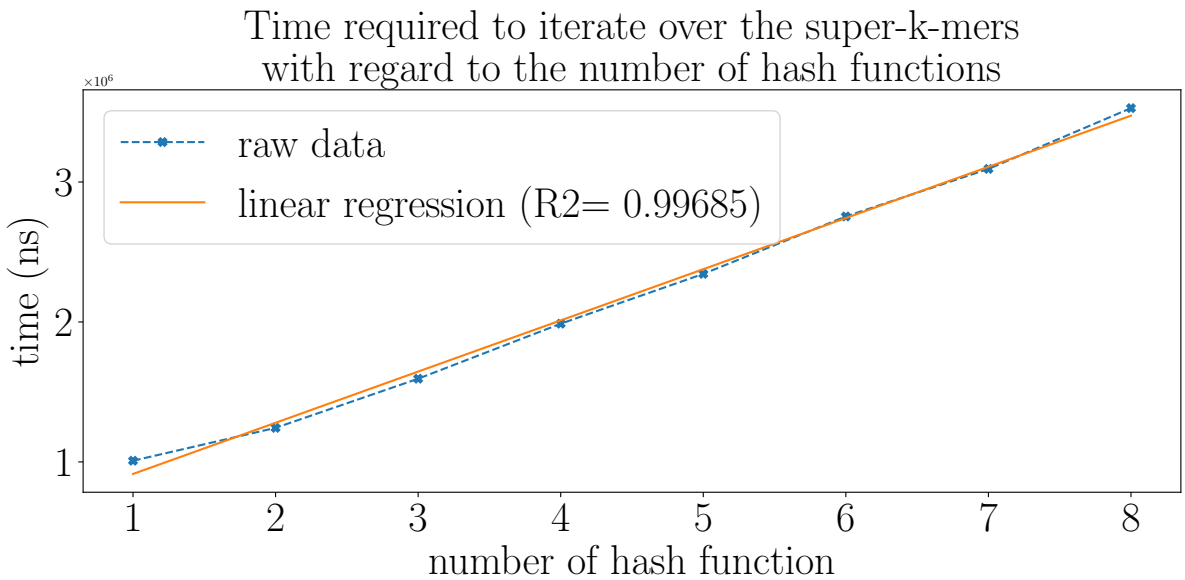


Fig. 12: Time taken to index a random string of size 10^5. The linear regression shows that the time taken to iterate over the multiminimizers is approximately the time needed to iterate over one random minimizer scheme times the number of hash functions.