S1 Analysis of Lexicographical Minimizers

In this section, we prove that the asymptotical optimality for the lexicographic minimizer with $k_0 = \lfloor \log_\sigma(w/2) \rfloor - 2$ can be used to derive the asymptotical optimality for other values of $k \geq \log_\sigma(w) - c$.

Lemma 7. *The lexicographic minimizer with $k = k_0 - c$ for constant $c \geq 0$ has density $O(1/w)$.*

Proof. We define W'^{+} and W'^{-} in the same way for this new minimizer with parameter k. We also call the new minimizer as k-minimizer, and the minimizer in Theorem 1 as k_0-minimizer.

We now claim $|W'^{+}| \leq |W^{+}|$. This is because given a window in W'^{+}, we can always append 0^c before the start of the window to get a window (for the k_0-minimizer) in W^{+}. Assume otherwise, we let $z_0 = 0^c \cdots$ be the k_0-mer at the start of the extended window and z_1 be the k_0-mer picked by the k_0-minimizer. This requires z_1 to also have form of $0^c \cdots$, and that the k-mer after 0^c is strictly smaller than the last k-mer of z_0. However, if this is the case, the original window will not be in W'^{+} as the first k-mer is not minimal.

Similarly, we claim $|W'^{-}| \leq |W^{-}|$ as given a window in W'^{-} we can append 0^c after the end of the window to get a window in W^{-}. Assume otherwise, we let $z_0 = \cdots 0^c$ be the last k_0-mer at the end of the extended window and z_1 be the k_0-mer picked by the k_0-minimizer. Since z_0 ends with 0^c and z_1 is smaller or equal to z_0, the k-prefix of z_1 must be smaller or equal to the k-prefix of z_0. This means the original window will not be in W'^{-} as there is a k-mer that is smaller or equal to the last k-mer.

Therefore $|W'^{+}| = O(\sigma^w)$ and $|W'^{-}| = O(\sigma^w)$, and the density is $\sigma(|W'^{+}| + |W'^{-}|)/\sigma^{w+k} = O(\sigma^{-k}) = O(1/w)$. □

On the other hand, for large k the following lemma establishes asymptotically optimal density. Recall $W^{\ddagger}$ is the set of windows (for the k_0-minimizer) such that the last k_0-mer is among the smallest, but not necessarily unique (so the minimizer might not always pick it).

Lemma 8. *The lexicographic minimizer with $k > k_0$ has density $O(1/w)$.*

Proof. We define W'^{+} and W'^{-} in the same way for this new minimizer, and again call the new minimizer the k-minimizer.

We first claim $|W'^{+}| \leq \sigma^{k-k_0}|W^{+}|$. This is because given a window in W'^{+}, we can remove the last $k - k_0$ bases to get a window in W^{+}. To see this, we only need to show for the shorter window, the first k_0-mer is less than or equal to every other k_0-mer. Since the original window is in W'^{+}, the first k-mer in the original window is less than or equal to every other k-mer. This means the k_0-long prefix of the first k-mer is also less or equal to the k_0-long prefix of every other k-mer in the original window, which is exactly the condition for the short window in W^{+}. For every window x in W^{+}, there are σ^{k-k_0} windows in Σ^{w+k-1} that x is a prefix of, which means $|W'^{+}| \leq \sigma^{k-k_0}|W^{+}|$.

We now claim $|W'^{-}| \leq \sigma^{k-k_0}|W^{\ddagger}|$. Similarly, given a window in W'^{-}, we can again remove the last $k - k_0$ bases to get a window in $W^{\ddagger}$. As the original window is in W'^{-}, the last k-mer in the original window is less than every other k-mer. This means the k_0-long prefix of the last k-mer is less or equal to the k_0-long prefix of every other k-mer in the original window. (Note that if a k-mer is less than another, their prefix can be equal.) The k_0-long prefixes of k-mers in the original window are exactly the k_0-mers of the shorter window, so the short window is in $W^{\ddagger}$. With a similar argument, we have $|W^{-}| \leq \sigma^{k-k_0}|W^{\ddagger}|$.

These two facts combined means we can calculate the density of the new minimizer as follows:

$$\begin{aligned}\sigma(|W'^{+}| + |W'^{-}|)/\sigma^{w+k} &\leq \sigma^{k-k_0+1}(|W^{+}| + |W^{\ddagger}|)/\sigma^{w+k} \\ &= (|W^{+}| + |W^{\ddagger}|)/\sigma^{w+k_0-1} \\ &= O(\sigma^{-k_0}) = O(1/w).\end{aligned}$$

□

S2 Analysis of UHS from Random Minimizers

We discuss the UHS with $w > k$ and $w < k$ separately. In the first case, the best UHS has its relative size lower bounded by the decycling set, and in the second case, a stronger bound of $1/w$ is available.

Lemma S14. *For sufficiently large k and $w > k - (3+\epsilon)\log_\sigma(k+1)$, there exists a UHS of relative size $2/k + o(1/k)$ with path length w. This is a $2 + o(1)$ approximation of the minimum size UHS.*

Proof. We pick $k' = (3 + \epsilon)\log_\sigma(k + 1)$ (ignoring rounding for simplicity), and let $w' = k - k'$. By our setup, $w \geq w'$, and $k' = o(k)$ (implying $w' = k - o(k)$). By Theorem 3, the charged context set of a random minimizer $(w', k', \mathcal{O}')$ has relative size $2/w' + o(1/w') = 2/k + o(1/k)$. By Theorem 4, the charged context set is a UHS over k-mers with path length w. This finishes the first part of the proof.

For the second part, by Lemma 2 the minimum size UHS, regardless of path length, will have relative size at least $1/k - o(1/k)$. As our proposed UHS has relative size $2/k + o(1/k)$, it is a $2 + o(1)$ approximation of the minimum size UHS. □

Lemma S15. *For sufficiently large k and $w \leq k - (3+\epsilon)\log_\sigma(k+1)$, there exists a UHS of relative size $2/w + o(1/w)$ with path length w. This is a $2 + o(1)$ approximation of the best possible UHS.*

Proof. We pick $w' = w$ and $k' = k - w$. By our setup, $k' \geq (3 + \epsilon)\log_\sigma(w' + 1)$ as $w < k$. By the identical argument as seen in the last lemma, we obtain a UHS over k-mers with path length w and relative size $2/w + o(1/w)$.

For the second part, note that the minimal relative size of a UHS with path length w is $1/w$ as it need to hit every one of every w k-mer on the de Bruijn sequence of order k, where every k-mer appears exactly once. As our set has relative size $2/w + o(1/w)$, it is a $2 + o(1)$ approximation. □

These two lemmas give us the desired result.

S3 More Analyses of the Miniception

This section contains analyses of the Miniception that are not included in the main text.

S3.1 Restricted Sampling

In this section, we formally prove the correctness of the restricted sampling process.

Lemma 12. *Denote the distribution generated by the restricted sampling process as $\mathcal{S}^{+}(x)$, then $\mathcal{S}^{+}(x) = \mathcal{R}^{+}(x)$.*

Proof. We prove the two distributions are identical in two parts. First, we show they have the same support (generate the same set of permutations). Note that any permutations sampled from $\mathcal{S}^{+}(x)$ satisfies E_0^{+} by the swapping process, and any permutation satisfying E_0^{+} can be sampled from $\mathcal{S}^{+}(x)$ as it can simply be the initial distribution and unchanged by the swapping process. Second, given $\mathcal{R}^+(x)$ is a uniform distribution over permutations satisfying E_0^+, we only need to prove for any two permutation satisfying E_0^+ they are generated by $\mathcal{S}^+(x)$ with the same probability. This is true because every permutation satisfying E_0^+ has exactly w preimages in the swapping process. □

Lemma 13. *Given an order of k_0-mers sampled from $\mathcal{S}^+(x)$, conditioned on the first k_0-mer being overall minimal, the k_0-mer order excluding the first k_0-mer follows the unrestricted distribution $\mathcal{R}(x)$.*

Proof. Our goal is to prove the two distributions are equal, and we will use the same two-step process. First, every permutation can be generated by both processes. Second, as proved before, $\mathcal{S}^+(x)$ generates each eligible permutation with identical probability, and this holds for the conditional distribution as each permutation has exactly one preimage. This means it is the same distribution as $\mathcal{R}(x)$ where each permutation is also generated with the same probability. □

S3.2 Analysis of $Q_n^-(x)$ with $x \leq 2$

Based on our analysis of $Q_n^+(x)$ for $x \leq 2$, as presented in the main text, we derive the recurrence for $Q_n^-(x)$ for $x \leq 2$ here.

To start with, we similarly define the restricted sampling process (where we swap the minimal element in the first w elements with the w^{th} element instead). We define $Q_n^-(x)$ to denote the same quantity as $Q_n^+(x)$, except the order is now sampled from the new restricted sampling process $\mathcal{S}^-(x)$. The derivation of $Q_n^-(x)$ is identical when the minimal k_0-mer in the sequence is not the w^{th} one. With probability $1/x$, the minimal k_0-mer will be the w^{th} one. In this case, if $x < 2$, again only one UHS k-mers will contain the minimal k_0-mer, but with $x = 2$ two k-mers will contain it (with $x = 2$, the k_0-mer is in the middle of the sequence, with at least $w - 1$ k_0-mers away from each end). All other UHS k-mers now come from the substring right to the minimal k_0-mer, whose relative length is now $x - 1$, and similar to our previous argument, the order within the substring follows $\mathcal{R}(x - 1)$. This yields the following recurrence:

$$Q_n^-(x) = \frac{1}{x}(P_{n-1}(x-1) + \int_1^x Q_{n-1}^-(t)\mathrm{d}t), n \geq 1, x < 2$$
$$Q_n^-(x) = \frac{1}{x}(P_{n-2}(x-1) + \int_1^x Q_{n-1}^-(t)\mathrm{d}t), n \geq 1, x = 2$$

S3.3 Analytical Solution for $P_n(x)$, $Q_n^+(x)$ and $Q_n^-(x)$ at $x \leq 2$

In this section, we provide the analytical solution to the integrals given $x \leq 2$, up to $n = 6$ as follows. The formula of $P_n(x)$ for $4 \leq n \leq 6$ is more complicated and is omitted here for clarity. We provide (in our Github repository) automatic symbolic integration codes that computes these integrals up to any specified n, and calculate $D(2)$ based on the integrals.

$$\begin{aligned}
P_0(x) &= 2/x - 1\\
P_1(x) &= -2 + 4\ln(x)/x + 2/x\\
P_2(x) &= -4 + 4\ln^2(x)/x + 4\ln(x)/x + 4/x\\
P_3(x) &= -8 + 8\ln^3(x)/3x + 4\ln^2(x)/x + 8\ln(x)/x + 8/x\\
Q_0^+(x) &= 0\\
Q_1^+(x) &= 2/x^2 - 1/x\\
Q_2^+(x) &= \ln(x)(4/x^2 - 1/x)\\
Q_3^+(x) &= \ln^2(x)(4/x^2 - 1/2x)\\
Q_4^+(x) &= \ln^3(x)(8/3x^2 - 1/6x)\\
Q_5^+(x) &= \ln^4(x)(4/3x^2 - 1/24x)\\
Q_6^+(x) &= \ln^5(x)(8/15x^2 - 1/120x)\\
Q_0^-(x) &= 0\\
Q_1^-(x) &= 1/x, x < 2; 0, x = 2\\
Q_2^-(x) &= \ln(x)/x, x < 2; (1 + \ln(x))/x, x = 2\\
Q_3^-(x) &= \ln^2(x)/2x\\
Q_4^-(x) &= \ln^3(x)/6x\\
Q_5^-(x) &= \ln^4(x)/24x\\
Q_6^-(x) &= \ln^5(x)/120x
\end{aligned}$$

S3.4 Deriving the Integrals for $x > 2$

To extend our analysis to other values of w, we will derive the recurrence formula for $P_n(x)$, $Q_n^+(x)$ and $Q_n^-(x)$ for $n \geq 2$. Note that as $w = w_0 + 1$ no longer holds, the relative length of a sequence is defined as the number of k_0-mers in the sequence divided by $w_0 + 1$, and $\mathcal{R}(x)$ is now a random permutation of $x(w_0 + 1)$ elements. Definition for $\mathcal{S}^{\pm}(x)$ and the desired quantities change accordingly.

We will follow the same general argument as before, by iterating over the location of minimal k_0-mer within the sequence as $t(w_0 + 1)$ for $0 \leq t \leq x$, and discuss the different scenarios. The extra consideration comes from the fact that now it is possible both left and right substrings can produce UHS k-mers, so we also need to iterate over the number of UHS k-mers from one substring. We will define the convolution operator to represent this iteration process:

$$[A(s) * B(t)]_n = \sum_{m=0}^{n} A_m(s)B_{n-m}(t)$$

We start with the derivation of $P_n(x)$ for $x > 2$. For $n = 0$ and $x > 2$, $P_n(x) = 0$ as wherever the minimal k_0-mer is, there will be one UHS k-mer containing it. For the rest, we define the middle region as the subregion that is at least w_0 k_0-mers away from both the first and the last k_0-mer in the sequence. Its relative length is $x - 2$. If the minimal k_0-mer falls in this region, two UHS k-mers (recall a k-mer consists of (w_0+1) k_0-mers) will contain the k_0-mer and both substrings split by the minimal k_0-mer may contain more UHS k-mers. If the minimal k_0-mer falls outside this region, the analysis is identical to the previous case. This yields the following recurrence:

$$\begin{aligned}P_n(x) &= \frac{1}{x}(2\int_0^1 P_{n-1}(x-t)\mathrm{d}t \\ &+ \int_1^{x-1}\sum_{m=0}^{n-2} P_m(t)P_{n-2-m}(x-t)\mathrm{d}t) \\ &= \frac{1}{x}(2\int_0^1 P_{n-1}(x-t)\mathrm{d}t \\ &+ \int_1^{x-1}[P(t) * P(x-t)]_{n-2}\mathrm{d}t)\end{aligned}$$

For $Q_n^+(x)$, we similarly define the middle region. If the minimal k_0-mer before swapping process falls to the left of the middle region (with probability $1/x$), it falls within the first k-mer and its order is swapped with the first k_0-mer. The probability of observing $n-1$ UHS k-mers outside the first k_0-mer is the same $P_{n-1}(x)$. If the minimal k_0-mer is to the right of the middle region, one UHS k-mer contains this k_0-mer and all other UHS k-mers comes from the left substring, following $\mathcal{S}^+(x-1)$. If the minimal k_0-mer is in the middle region, two UHS k-mers contain the k_0-mer and both substrings may contain more UHS k-mers, with the order in the left substring conditioned on $\mathcal{S}^+(t)$ and the order in the right substring conditioned on $\mathcal{R}(x-t)$. This yields the following recurrence:

$$Q_n^+(x) = \frac{1}{x}(P_{n-1}(x) + \int_1^{x-1}[Q^+(t) * P(x-t)]_{n-2}\mathrm{d}t + \int_{x-1}^{x} Q_{n-1}^+(t)\mathrm{d}t)$$

The derivation for $Q_n^-(x)$ is extremely similar to that of $Q_n^+(x)$. Note that now $x \geq 2$, two UHS k-mers are guaranteed when the minimal k_0-mer is the w^{th} one.

$$Q_n^-(x) = \frac{1}{x}(P_{n-2}(x-1) + \int_1^{x-1}[Q^-(t) * P(x-t)]_{n-2}\mathrm{d}t + \int_{x-1}^{x} Q_{n-1}^-(t)\mathrm{d}t)$$

Following the methods of truncating distributions outlined in the main text, noting that now $P(E_0^+) = P(E_0^-) = 1/(w_0+1) = (x-1)/w$, we can calculate density factor bound as follows:

$$\begin{aligned}M_i &= (Q_i^+(x) + Q_i^-(x))/2 \\ D(x) &= 4(x-1)(\sum_{i=1}^{n} M_i/i + (1-\sum_{i=1}^{n} M_i)/(n+1))\end{aligned}$$

, assuming the integrals are derived up to n UHS k-mers. The final density bound will be $D(x)/w + o(1/w)$.

S3.5 Estimating $D(x)$ with Dynamic Programming

As $D(x)$ represents the density of the Miniception when $w \approx (x-1)k$, with $k \to \infty$, it is natural to approximate $D(x)$ by selecting a large value of k and calculate the density of the Miniception with $w = (x-1)k$. Here we make the assumption that all k_0-mers in the string are unique and $k_0 \ll k$, so we simply take $k_0 = 1, w = (x-1)k$ and assume the order of the k_0-mers in the context follows $\mathcal{R}(x)$. By our analysis for the integral method (See Section S3.6), the final density bound will be off by $O(1/k)$ compared to the integrals, so by choosing k to be large enough we can ensure the derived bound is close to the integral bound, which again by Section S3.6 approximates the behavior of the Miniception for arbitrary large values of k.

Now, let $P[l,n]$ be the probability that given permutation of l k_0-mers, there are exactly n UHS k-mers. Recall a k-mer is a UHS k-mer if its first or last k_0-mer is the smallest in the substring, which is of length k now. We again enumerate over the location of the smallest k_0-mer in permutation, and denote this value i. Each specific location is the minimum with probability $1/l$. If $i \geq k-1$, the k-mer that ends at i (meaning its last k_0-mer is this one) is a UHS k-mer. By symmetry, if $i \leq l-k$, the k-mer starting at i is a UHS k-mer. The rest of UHS k-mers will come from the two substrings, obtained by removing the i^{th} k_0-mer from the sequence, and as before, we need to enumerate the number of UHS k-mers from either substrings. This results in the following recurrence, where we use $f_{l,n}(i)$ to denote the number of UHS k-mers that need to come from either substrings:

$$\begin{aligned}(A[x] * B[y])_n &= \sum_{i=0}^{l} A[x,i]B[y,n-i] \\ f_{l,n}(i) &= n - \mathbf{1}(i \geq k-1) - \mathbf{1}(i \leq l-k) \\ P[l,n] &= \frac{1}{l}\sum_{i=0}^{l-1}(P[i] * P[l-1-i])_{f_{l,n}(i)}\end{aligned}$$

With the dynamic programming, we only need to provide that $P[i,0] = 1, P[i,n] = 0$ for $i < k$ and $n \geq 1$. We can then calculate the values of $P[l,n]$ up to $l = xk$ and some preset value of n with the requirement $n > 2x$.

We can similarly generate the recurrence relationship for $Q^+[l,n]$ and $Q^-[l,n]$ (derivations are highly similar and omitted here):

$$\begin{aligned}Q^+[l,n] &= \frac{1}{l}(kP[l-1,n-1] \\ &+ \sum_{i=k}^{l-1}(Q^+[i] * P[l-1-i])_{f_{l,n}(i)}) \\ Q^-[l,n] &= \frac{1}{l}(kP[l-k, n-1-\mathbf{1}(l \geq 2k-1)] \\ &+ \sum_{i=k}^{l-1}(Q^-[i] * P[l-1-i])_{f_{l,n}(i)})\end{aligned}$$

The approximate density factor bound (we use the density factor bound for numerical stability and ease of comparison), which we denote as $D_k(x)$, can be calculated as follows using our previous argument of truncating the distribution.

$$\begin{aligned}M_i' &= (Q^+[xk,i] + Q^-[xk,i])/2 \\ D_k(x) &= 4(x-1)(\sum_{i=1}^{n} M_i'/i + (1-\sum_{i=1}^{n} M_i')/(n+1))\end{aligned}$$

The final density bound for the Miniception under this configuration will be $D_k(x)/w + o(1/w)$, where the $o(1/w)$ term comes from the event that some k_0-mers in a context can be identical.

S3.6 Error Analysis for the Integral Method

In the derivation for the Miniception, we assume $k \to \infty$ and use an integral instead of summation for derivation of $P_n(x)$ and other quantities.

In this section, we show the error introduced by this method is small enough to not affect the final result asymptotically. Specifically, we show the following:

Lemma S16. *Let $D(x)$ be the density factor bound derived from the integrals, and $D_k(x)$ be the density factor bound calculated from the dynamic programming with window length $(x-1)k$ and k-mer length k (measured in number of k_0-mers), assuming all k_0-mers are distinct. We have $|D(x) - D_k(x)| = O(nx/k)$, where n is the largest number of UHS k-mers considered by both processes.*

As shown in the main text, this lemma serves a dual purpose. It means $D(x)$ is a good approximation to any $D_k(x)$ up to asymptotically negligible errors, and by calculating $D_k(x)$ for sufficiently large k, we can estimate $D(x)$ accurately which in turn approximates other $D_k(x)$. In other words, $D(x)$ serves as a bridge to connect the density factor bound $D_k(x)$ for different values of k, which then bound the density of the actual minimizer as long as k_0 satisfies the condition for Lemma 8. Note that in derivation of $D_k(x)$ the concept of k_0-mers are already abstracted away, and for simplicity we assume $k_0 = 1$ in that process, meaning every k-mer now consists of k k_0-mers.

Recall in the main text we set up the integrals to derive the distribution of m_{total} conditioned on first k-mer being a UHS k-mer. Following the convention in the dynamic programming process, we assume $k_0 = 1$, meaning $k = w_0 + 1$. The context has exactly xk k_0-mers, and each k-mer is of length k. Note that all lengths in the section are measured in k_0-mers. As shown in the main text, conditioned on E_0, the context will contain at least two UHS k-mers.

Recall that we derive the values of $P_n(x)$ up to n. An alternative view of the integral recurrence is that we determine at most $n+1$ locations that contains the minimal k_0-mers in a certain substring, and calculate the number of UHS k-mers based on the order and the distance of these locations. For example with $x = 2$, if the first location (the minimal k_0-mer in the whole context) is on the left half of the context, the second location is for the minimal k_0-mer in the right substring. If the second location is in the middle region of that substring, we determined the number of UHS k-mers (1 in this case) using two locations. If instead, the second location is not inside the middle region, a third location is determined inside the sub-sub-sequence that could still generate a UHS k-mer, and these three locations can collectively determine how many UHS k-mers are in the whole context. This process continues until no substrings can generate more UHS k-mers, or $n+1$ UHS k-mers have been generated. There are only $n+1$ determined locations total, as for each recursion there is one corresponding UHS k-mer, either right before the start of the substring or right after the end of the substring. This is similar when we derive the values of $Q_n^+(x)$ and $Q_n^-(x)$, where we determine the location of the minimal k_0-mer before swapping.

We denote the sequential decision process to determine number of UHS k-mers, conditioned on E_0^+ (corresponding to $Q_n^+(x)$) and E_0^- (corresponding to $Q_n^-(x)$), as $m \sim \mathcal{P}_n^+(x)$ and $m \sim \mathcal{P}_n^-(x)$, respectively. Formally:

$$P_{m\sim\mathcal{P}^+(x)}(m = i) = Q_i^+(x), i \le n$$

$$P_{m\sim\mathcal{P}^+(x)}(m = n+1) = 1 - \sum_{i=1}^{n} Q_i^+(x)$$

And this is symmetric for $\mathcal{P}^-(x)$. With this definition, we have:

$$D(x) = 2(x-1)(\mathbb{E}_{m\sim\mathcal{P}^+(x)}(1/m) + \mathbb{E}_{m\sim\mathcal{P}^-(x)}(1/m))$$

Similarly, we can propose an alternative view of the dynamic programming, where we determine at most $n+1$ locations that contains the minimal k_0-mers in a certain sequence. The difference is that in the previous cases, the locations are picked on a real axis $[0, x]$, but now it is picked from a discrete set of locations in $[xk]$. We will now argue the two processes are not that different.

We first establish a mapping $M(t) : \mathbb{R} \to [xk]$ from real numbers to discrete locations, by multiplying the real number by k and rounding down. When a real number is sampled uniformly from $[0, x]$, its mapping will be a uniform distribution over $[xk]$ (that is, $1/xk$ chance to map into each location). Now, we consider emulating the decision process from the dynamic programming (which we denote $\mathcal{P}_k^+(x)$, for simplicity). Each time a location is randomly sampled (the location of minimal k_0-mer in current substring of consideration) from $\mathcal{P}^+(x)$, we emulate one step of $\mathcal{P}_k^+(x)$ by selecting the mapped discrete location as the location of the minimal k_0-mer. The process continues until the end of decision process.

We now examine the emulation process in more detail. At each step of the decision process, we sample the location of the minimal k_0-mer in current substring of consideration, before the swapping process if the substring of consideration is still conditioned on E_0^+ or E_0^-. The boundary of the substring is determined by two previous samples (or simply the boundary), which we denote L and R. The discrete decision process we are emulating will be sampling the location of the minimal k_0-mer in current substring of consideration, which will be the substring from $M(L)$ to $M(R)$, including neither ends. It can be seen that we are sampling from a slightly longer range in the original process. As long as our sampled location T satisfies $M(L) < M(T) < M(R)$, the distribution of $M(T)$ is uniform over possible selections. If $M(T) = M(L)$ or $M(T) = M(R)$ at any step of the emulation, we call the whole process failed and do not continue. The swapping process can be addressed similarly, as in this case $L = 0$ is guaranteed, and the emulated and original decision process will never disagree on whether the swap should be triggered.

Assume $M(T)$ passes this check, meaning it is uniformly sampled from possible locations between $M(L)$ and $M(R)$. We will now decide on the next step in the process. This includes two parts: Counting number of UHS k-mers including the minimal k_0-mer, and determining if a substring can still produce more UHS k-mers. Both parts involves two distance checks: If $T - L > 1$ in the original decision process, and if $M(T) - M(L) \ge w_0$ in the discrete decision process, for checking if the k-mer ending with the minimal k_0-mer in this substring constitutes a valid UHS k-mer. If the original decision process and the emulated decision process will give different answer to this check, we call the whole process failed and do not continue. For deciding if the left substring can produce more UHS k-mers, we replace w_0 with $w_0 + 1$ and the analysis is the same. It is also symmetric for the right substring and checking the k-mer starting with minimal k_0-mer.

Now, if a decision process does not fail (by either of the previously defined criteria), we say it perfectly emulates a discrete decision process. By the way it is defined, the emulated process randomizes the location of minimal k_0-mers correctly, make the same decision as the original process at every step, and count the same number of UHS k-mers. We now characterize these "safe decision processes" with the following notation. Let $\mathcal{S}_k^+$ denote a distribution whose support is $\{\times, 1, 2, 3, \cdots, n+1\}$ derived from the aforementioned emulation process. $\times$ denotes a failed process, and an integer denotes the number of UHS k-mers determined from the process. We claim $P_{m\sim\mathcal{S}_k^+}(m = i) \le P_{m\sim\mathcal{P}^+}(m = i)$ and $P_{m\sim\mathcal{S}_k^+}(m = i) \le P_{m\sim\mathcal{P}_k^+}(m = i)$ simultaneously for all $i \in \{1, 2, 3, \cdots, n+1\}$. Intuitively, the distribution $\mathcal{P}^+$ and $\mathcal{P}_k^+$ looks the same if the emulation is successful.

The first part of the statement can be proved by the definition of the emulation process: $P_{m\sim\mathcal{S}_k^+}(m = i)$ is the probability that the emulation successes and returns i UHS k-mers, which is never greater than the probability of i UHS k-mers regardless of the emulation outcome

($P(A \cap B) \leq P(A)$). The second part is can be proved by viewing the emulation from the perspective of the discrete process (which we call a reverse emulation, however they are in fact the same process): At each step of the discrete decision process, we first randomly select the location of the minimal k_0-mer, and randomly choose a real number that maps to this location as its real coordinate. We then do a failure check by rolling a random number inside $[L, R]$ (from the real number locations selected in previous steps), and fail the process if the rolled location collide. Similarly, at each distance check we first get the outcome from discrete locations, then fail the process if the real number locations disagree with the outcome. In this perspective, we can prove the second part of the statement in the same way: $P_{m\sim\mathcal{S}_k^+}(m=i)$ is the probability that this reverse emulation successes and returns i UHS k-mers, and is never greater than the probability of getting i UHS k-mers regardless of the outcome of reverse emulation.

We now bound the probability of failure, that is $P_{m\sim\mathcal{S}_k^+}(m=\times)$. As the decision process consists of $n+1$ steps, we will bound the probability at each step. At each step, the interval for random selection has at least relative length of 1 (otherwise no UHS k-mers will be from this section), and there are at least k k_0-mers for selection. The key observation here is there are only 6 ways to fail (colliding with $M(L)$, disagreement on whether left substring contains more UHS k-mers, disagreement on whether the k-mer ending at smallest k_0-mer is valid UHS k_0-mer, and their symmetric counterparts), and for each of them, there is a interval of length at most $1/k$ such that the fail event is triggered if and only if the minimal k_0-mer is inside the interval. In other words, the probability to fail at each decision step is at most $6/k = O(1/k)$, and the probability of failure at any step is $O(n/k)$.

We now have the tools to bound $|D(x) - D_k(x)|$. We first bound the term $\mathbb{E}_{m\sim\mathcal{P}^+(x)}(1/m)$. We denote S as the event that the emulation is successful, and $\mathcal{F}^+(x)$ as the distribution of UHS k-mer count when the emulation fails.

$$\begin{aligned}\mathbb{E}_{m\sim\mathcal{P}^+(x)}(1/m) &= \sum_{i=1}^{n+1} P_{m\sim\mathcal{P}^+(x)}(m=i)/i \\ &= P(S)\sum_{i=1}^{n+1} P_{m\sim\mathcal{S}_k^+(x)}(m=i)/i \\ &+ P(\bar{S})\sum_{i=1}^{n+1} P_{m\sim\mathcal{F}^+(x)}(m=i)/i\end{aligned}$$

And similarly, let $\mathcal{F}_k^+(x)$ as the distribution of UHS k-mer count when the reverse emulation fails.

$$\begin{aligned}\mathbb{E}_{m\sim\mathcal{P}_k^+(x)}(1/m) &= \sum_{i=1}^{n+1} P_{m\sim\mathcal{P}_k^+(x)}(m=i)/i \\ &= P(S)\sum_{i=1}^{n+1} P_{m\sim\mathcal{S}_k^+(x)}(m=i)/i \\ &+ P(\bar{S})\sum_{i=1}^{n+1} P_{m\sim\mathcal{F}_k^+(x)}(m=i)/i\end{aligned}$$

This leads to the following:

$$\begin{aligned}&|\mathbb{E}_{m\sim\mathcal{P}^+(x)}(1/m) - \mathbb{E}_{m\sim\mathcal{P}_k^+(x)}(1/m)| \\ &= P(S)|\sum_{i=1}^{n+1}(P_{m\sim\mathcal{F}^+(x)}(m=i) - P_{m\sim\mathcal{F}_k^+(x)}(m=i))/i| \\ &\leq P(S) = O(n/k)\end{aligned}$$

Substitute this back, we get $|D(x) - D_k(x)| = O(nx/k)$.

Note that this bound is not tight. This is because we overestimate the difference term $\sum_{i=1}^{n+1} P_{m\sim\mathcal{S}_k^+(x)}(m=i)/i$ by simply upper bounding it by 1. However, for larger value of k and x, the context is almost guaranteed to have around $2x$ UHS k-mers, and the difference here is more on the order of $O(1/x^2)$. This means in practice the bound is more like $O(1/k)$ (considering $n = \Theta(x)$ is a reasonable choice) with the possibility of even smaller (as we do not consider that positive terms and negative terms cancel out each other), which also explains why we can pick $k = 2500$ in our simulations.

S4 Implementing the Miniception

The implementation takes a sequence S as input (as well as other parameters specifying the minimizer), and returns the list of picked locations. We will derive a linear time implementation, meaning the algorithm runs in time $O(k|S|)$ as it takes $O(k)$ time just to process and compare k-mers. We will discuss our algorithms based on the assumption that a k-mer fits in a word (This means $k \leq 32$ for 64-bit systems), Value of k beyond this limit is rare in practice, and our algorithm also easily adapts to general values of k.

While the Miniception is defined with many randomness, implementing it means sticking to one "instantiation", and the orders $\mathcal{O}_0$ and $\mathcal{O}$ are fixed. We will assume the alphabet is $\{0, 1, 2, 3\}$ so a k-mer can be written as an integer in $[4^k]$. We assume constant time access to $\mathcal{O}_0$ as a function $f_0 : [4^{k_0}] \to \mathbb{R}$, such that $x_0 < x_1$ in $\mathcal{O}_0$ if and only if $f_0(x_0) < f_0(x_1)$, and similarly a function $f : [4^k] \to \mathbb{R}$ for some order function that agrees with $\mathcal{O}$ when restricted to $\mathcal{C}_0$, meaning both f_0 and f can simply be a random hash function.

From the construction of the Miniception, we can recover $\mathcal{C}_0$ from $\mathcal{O}_0$. In most cases, this means we only need to store a small minimizer to implement the Miniception on the fly. In comparison, existing methods with precomputed UHS requires storage of the whole set.

Implementation of minimizers usually use monotone queues for linear time complexity, under the same assumption. The monotone queue is a special kind of deque that ensures the item in the queue are both in nondecreasing order and are inserted recent enough. Such data structure allows solving the problem of finding minimum element in every T-long window in input linear time, which is exactly the problem of identifying picked k-mers for a given sequence, as the picked locations are simply locations that are minimal in a w-long window. We will provide pseudocode for its implementation later this section.

As the Miniception consists of two minimizers, we use two monotone queues, one implementing the seed minimizer and one implementing the actual minimizer which identifies picked locations. As we only care whether a context is charged for the seed minimizer, the first monotone queue has expiry time w_0+1. The second monotone queue is implemented as in a normal minimizer.

We provide a pseudocode for the algorithm below. For simplicity, we will ignore the warmup part (before the first full window). We provide a reference implementation written in Python in our github repo. We start with the monotone queue, which as described can be considered as a deque with an extra parameter T (expiry time) and additional operations.

Algorithm 1 The Monotone Queue with Expiry Time T

```
procedure SetTime($t$)                          ▷ this procedure removes expired items
    while Queue is not empty do
        $x', t'$ ← front of queue
        if $t' \leq t - T$ then
            Pop $x', t'$ from queue
        else
            return
        end if
    end while
end procedure
procedure Insert($x$, $t$)                      ▷ input is (item, time) pair
    Set current time to $t$
    while Queue is not empty do
        $x', t'$ ← end of queue
        if $x < x'$ then                        ▷ minimizers prefer leftmost $k$-mers
            Pop $x', t'$ from queue
        else
            break
        end if
    end while
    Append $x, t$ to the end of queue
end procedure
```

The monotone queue achieves $O(k)$ amortized time complexity (recall each item in the queue is a k-mer), as each item will only be inserted and popped once. With the monotone queue, we can implement the Miniception. As mentioned before, we assume each k-mer can be held in a word.

Algorithm 2 The Miniception

```
procedure MiniceptionPicks($S$)
    $Q_0 = \text{MonoQueue}(w_0 + 1)$               ▷ $w_0 + 1$ is the window length
    $Q = \text{MonoQueue}(w)$
    for $i = (w + k - 1)$ to $|S|$ do              ▷ Setups ignored for clarity
        $m_0 \leftarrow (m_0 \times 4 + S[i]) \mod 4^{k_0}$     ▷ Latest $k_0$-mer
        $m \leftarrow (m \times 4 + S[i]) \mod 4^k$           ▷ Latest $k$-mer
        Insert $(f_0(m_0), i)$ to $Q_0$
        $t \leftarrow$ time of $Q_0[0]$              ▷ for current minimum $k_0$-mer in window
        if $(t = i)$ or $(t = (i - w_0))$ then
            Insert $(f(m), i)$ to $Q$                ▷ latest $k$-mer is in $\mathcal{C}_0$
        else
            Set current time to $i$ in $Q$           ▷ remove out-of-window $k$-mer
        end if
        $p \leftarrow$ time of $Q[0]$                ▷ this is the end of picked $k$-mer in window
        Pick $p - (k - 1)$ if not picked already
    end for
end procedure
```

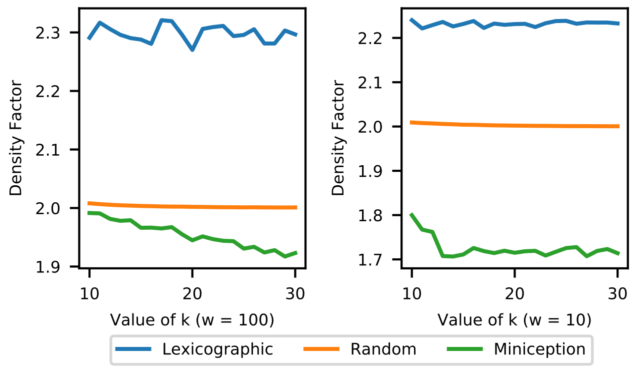

Fig. 7. Comparing density of the Miniception against lexicographic and random minimizers. We experiment with $w = 10$ (left half) and $w = 100$ (right half).

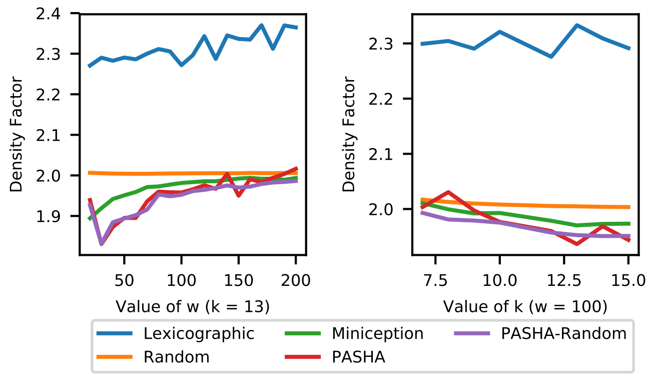

Fig. 8. Density factor comparison, now also with minimizers derived from PASHA outputs. Left half: experiments with fixed $k = 13$ and varying w. Right half: experiments with fixed $w = 100$ and varying k.

S5 Evaluating Minimizers on Human Reference Genome

We use the identical setup as in the main text, except the contexts are now sampled from hg38 human reference genome. As observed by Roberts *et al.* (2004b,a), order the alphabet by reverse frequency may improve performance of the lexicographic minimizer. For this reason, we use the order $C < G < A < T$ for all minimizers. We also consider two strategies for constructing a compatible minimizer given the UHS from PASHA. The authors suggested a lexicographic order within the UHS, but we also implement a minimizer where the order within the UHS is randomized, like the Miniception. Refer to the main text for more details.