Efficient minimizer orders for large values of k using minimum decycling sets

David Pellow[1], Lianrong Pu[1], Baris Ekim[2], Lior Kotlar[3], Bonnie Berger[2], Ron Shamir[1], and Yaron Orenstein[4,5]

[1] Blavatnik School of Computer Science, Tel-Aviv University, Israel
[2] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA
[3] Department of Computer Science, Ben-Gurion University, Israel
[4] Department of Computer Science, Bar-Ilan University, Israel
[5] The Mina and Everard Goodman Faculty of Life Sciences, Bar-Ilan University, Israel

Abstract. Minimizers are ubiquitously used in data structures and algorithms for efficient searching, mapping, and indexing of high-throughput DNA sequencing data. Minimizer schemes select a minimum k-mer in every L-long sub-sequence of the target sequence, where minimality is with respect to a predefined k-mer order. Commonly used minimizer orders select more k-mers overall than necessary and therefore provide limited improvement to runtime and memory usage of downstream analysis tasks. The recently introduced universal k-mer hitting sets produce minimizer orders resulting in fewer selected k-mers. Unfortunately, generating compact universal k-mer hitting sets is currently infeasible for $k > 13$, and thus cannot help in the many applications that need minimizers of larger k.

Here, we close this gap by introducing *decycling set-based minimizer orders*. We define new orders based on minimum decycling sets, which are guaranteed to hit any infinitely long sequence. We show that in practice these new minimizer orders select a number of k-mers comparable to that of minimizer orders based on universal k-mer hitting sets, and can also scale up to larger k. Furthermore, we developed a query method that avoids the need to keep the k-mers of a decycling set in memory, which enables the use of these minimizer orders for any value of k. We expect the new decycling set-based minimizer orders to improve the runtime and memory usage of algorithms and data structures in high-throughput DNA sequencing analysis.

Keywords: minimizers · de Bruijn graph · high-throughput sequencing · universal k-mer hitting set

1 Introduction

As the number and depth of high-throughput sequencing experiments grows, efficient methods to map, store, and search DNA sequences have become critical in their analysis. Sequence sketching is a fundamental building block of many of the basic sequence analysis tasks, such as assembly [20,4], alignment [22,19,11], and binning [2,1,6]. The common principle in all sketching techniques is the selection of a k-mer representative from a long DNA sequence for indexing sequences in data structures or algorithms. Key parameters for evaluating the merits of sketching techniques are density [13], defined as the fraction of k-mers selected, maximum bin load [12], defined as the maximum number of windows sketched by a single k-mer, and conservation of the sketch under mutations or sequencing errors.

One of the most common sequence sketching techniques is minimizers [23]. The minimizer of an L-long sequence is the minimum among all the $w = L-k+1$ k-mers that it contains, according to some order o over the k-mers. Selecting the minimizers from all overlapping L-long windows of a sequence provides a sketch of that sequence. Despite the advantages of minimizers, commonly used minimizer orders, such as lexicographic and random, have been shown to perform poorly in density [13] and maximum bin load [6]. In addition, the conservation under mutations and robustness to errors of such minimizer orders is low relative to other sketching techniques [3].

A recent breakthrough in developing minimizer orders with lower density has been achieved by compact universal k-mer hitting sets (UHSs) [18]. A UHS is a set of k-mers guaranteed to hit any L-long sequence. In terms of a complete de Bruijn graph of order k, a minimum UHS is a minimum set of nodes whose removal leaves no path of length $L - k + 1$ in the graph. Heuristic algorithms for finding a minimum UHS include DOCKS [18] and PASHA [5], both of which approach UHS construction as a path covering problem in a complete de Bruijn graph. The UHS-based minimizer orders were shown to achieve lower density than common orders [13,5]. However, constructing and storing UHSs is inefficient due to the exponential dependence of the heuristic algorithms on k, and currently compact UHSs are available only for $k \leq 13$. Key to these methods is the ability to efficiently identify a minimum decycling set, which is a set of k-mers guaranteed to hit any infinitely long sequence. A minimum decycling set can be generated in time log-linear in the de Bruijn graph size [15].

Partly due to the challenges in constructing UHSs, other recent works have focused on developing sequence-specific minimizer orders. For example, sequence-specific minimizer orders have been used in binning applications to achieve lower maximum bin size or more balanced bins than general minimizers [1,6]. Hoang et al. [8] used deep learning to achieve sequence-specific low-density minimizers for much longer k (up to 320). Still, these solutions are tailored to a specific sequence, and cannot be generally applied.

In this work, we developed new methods to construct general minimizer orders that scale to larger k. We defined minimizer orders based only on a minimum decycling set. We further improved the scalability of our approach by implementing an efficient method to query in linear time if a k-mer belongs to a minimum decycling set without the need to construct, store, or query the whole set. Finally, we demonstrate that our new decycling-set-based minimizer orders achieve density that is comparable to or better than UHS-based orders. The minimizer orders we defined thus provide for the first time general orders with low density that can scale to any value of k. All code developed under this project is publicly available via github.com/OrensteinLab/DecyclingSetBasedMinimizerOrder.

2 Preliminaries and definitions

We begin by defining and providing theoretical background on concepts necessary for the description and evaluation of our methods.

k-mer: For a string S over an alphabet Σ, a *k-mer* is a contiguous substring of length k. We denote the k-mer starting at position i as $S[i, i+k-1]$.

k-mer order: For a function on k-mers $o : \Sigma^k \rightarrow \mathbb{R}$, we say that k-mer x_1 is less than x_2 under o ($x_1 <_o x_2$) iff $o(x_1) < o(x_2)$.

Minimizer scheme: A *minimizer scheme* is a function $f_{k,w,o} : \Sigma^{w+k-1} \rightarrow \{0, \dots, w-1\}$. Function f returns the position of the minimum k-mer in a given window of w overlapping k-mers (i.e., in every $L = w + k - 1$ long window). By convention, ties are broken by choosing the left-most k-mer. The *minimizers* of a string S, denoted as $\mathcal{M}_{k,w,o}(S)$, are all the positions in the string that are selected by applying the scheme to all overlapping L-long windows of S:

$$\mathcal{M}_{k,w,o}(S) = \bigcup_{j=0}^{|S|-w-k+1} \left\{ \operatorname*{argmin}_{o \atop i:i\in[j,j+w-1]} S[i, i+k-1] \right\}$$

Universal hitting set (UHS): A *universal hitting set* $\mathcal{U}_{k,L} \subseteq \Sigma^k$ is a set of k-mers such that any L-long string contains at least one k-mer from $\mathcal{U}_{k,L}$ as a contiguous substring. By construction, at least one k-mer from $\mathcal{U}_{k,L}$ must appear in every window of $w = L - k + 1$ overlapping k-mers, and thus it is possible to define minimizer orders that are compatible with a UHS. An order $o_{\mathcal{U}_{k,L},h}$ is *compatible* with $\mathcal{U}_{k,L}$ if for $x_1 \in \mathcal{U}_{k,L}, x_2 \notin \mathcal{U}_{k,L} \implies x_1 <_{\mathcal{U}_{k,L},h} x_2$, and otherwise, when x_1 and x_2 are either both in or both not in the UHS, then $x_1 <_{\mathcal{U}_{k,L},h} x_2 \iff h(x_1) < h(x_2)$ for some order h.

Partition-compatible minimizer order: We extend the above definition of UHS-compatible orders to minimizer orders that are compatible with an ordered partition of Σ^k. Given an ordered partition of Σ^k, $\Pi = [\mathcal{C}_1, \dots, \mathcal{C}_m]$, we define a compatible minimizer order $o_{\Pi,h}$ such that for $x_1 \in \mathcal{C}_i$, $x_2 \in \mathcal{C}_j$ $i < j \implies x_1 <_{\Pi,h} x_2$ and if $i = j$ then $x_1 <_{\Pi,h} x_2 \iff h(x_1) < h(x_2)$ for some order h.

de Bruijn graph: A *de Bruijn graph* (dBG) of order k is a directed graph in which every node is labelled with a distinct k-mer and there may be a directed edge from node a to b iff the $(k-1)$-long suffix of a is the same as the $(k-1)$-long prefix of b. The edge is labelled with the $(k+1)$-long merge of the two labels. A *complete dBG* has a node for every possible k-mer and an edge for every possible $(k+1)$-mer. Paths in a dBG of order k represent sequences, and a path of w nodes represents a sequence of w overlapping k-mers. Thus, the nodes represented by a UHS $\mathcal{U}_{k,L}$ will be a covering set for all $(L-k+1)$-long paths in a complete dBG of order k.

Decycling set A *decycling set* in a graph $G=(V,E)$ is a set of nodes whose removal results in an acyclic graph. Finding a minimum decycling set (also called feedback vertex set) in an arbitrary graph is NP-hard [9]. We are interested in a minimum decycling set in a complete dBG of order k, which we denote by $\mathcal{D}_k$. Mykkeltveit [15] gave an efficient algorithm to construct such a set in time log-linear in the complete dBG size. A *pure cycle* is a set of nodes corresponding to all the cyclic rotations of some k-mer [15]. Mykkeltveit showed that $\mathcal{D}_k$ contains a single node from each pure cycle in a complete dBG. Moreover, each pure cycle defines a conjugacy class, and thus the pure cycles factor the complete dBG, namely every k-mer belongs to exactly one of the pure cycles.

Mykkeltveit embedding: To determine which of the cyclic rotations of a k-mer to include in $\mathcal{D}_k$, Mykkeltveit defined an embedding of k-mers in the complex plane. For a k-mer x, $\mathbb{M}(x)=(\mathbb{R}(x),\mathbb{I}(x))=\left(\sum_{i=0}^{k-1} x_i \sin\left(\frac{2\pi i}{k}\right), \sum_{i=0}^{k-1} x_i \cos\left(\frac{2\pi i}{k}\right)\right)$, where x_i is the numeric encoding of the i-th character of x (in our case the encoding of the DNA alphabet is: A=0, C=1, G=2, T=3) (Figure 1). The minimum decycling set constructed by Mykkeltveit's algorithm includes for each conjugacy class the first counter-clockwise rotation x such that $\mathbb{R}(x)>0$. When all rotations have $\mathbb{R}(x)=0$, any arbitrary k-mer from the cycle can be selected.

Mykkeltviet's algorithm has an efficient implementation due to Knuth [10]. This implementation uses the FKM algorithm [7] to enumerate the k-mer conjugacy classes in lexicographic order. The representative selected for each class is first one with $\mathbb{R}(x)>0$, and for classes with $\mathbb{R}(x)=0$ for all k-mer rotations, the lexicographically smallest k-mer is included in the decycling set. A minimum decycling set consists of $O(|\Sigma|^k/k)$ k-mers and it can be generated in time $O(k|\Sigma|^k)$, i.e. log-linear in the dBG size [15,21].

Minimizer density: The *expected density* of a minimizer scheme is the fraction of k-mer positions that will be selected as minimizers in expectation over an infinitely long random i.i.d. sequence. The *particular density* of a minimizer scheme on a specific sequence S (e.g., the human genome) is the fraction of k-mer positions selected by the scheme on that sequence.

$$density(S)=\frac{|\mathcal{M}_{k,w,o}(S)|}{|S|-k+1}$$

The *density factor* normalizes density for the window size w of the scheme. We follow the definition of Zheng et al. [8]: for a sequence S the *density factor* is $df(S)=\frac{|\mathcal{M}_{k,w,o}(S)|}{|S|-L+1}\cdot(w+1)$, where $L=w+k-1$. This definition of the density

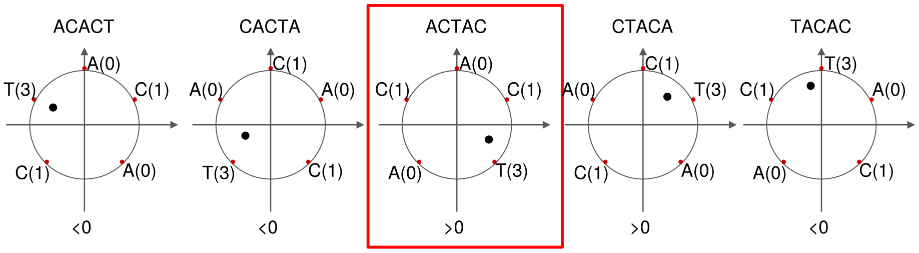

Fig. 1: **Mykkeltveit embedding.** The embedding is shown for the rotations of the k-mer ACACT, indicated above each subfigure. Each letter of the k-mer encodes a weight (in parentheses) placed at the k-th roots of unity (red dots). The embedding represents the center of mass of the k-mer (black dot). The sign of each embedding projected onto the real axis is shown below each rotation. In this example, ACTAC (red box) is the first counter-clockwise rotation x with $\mathbb{R}(x) > 0$, and is thus selected by Mykkeltveit's algorithm to participate in a minimum decycling set.

factor removes the dependence on L, e.g. making the expected density factor of all random minimizers the same, regardless of k and L. Note that other works define the density factor simply as the density times a factor of $(w+1)$ (c.f. [13]). Expected and particular density factors are defined analogously to expected and particular density. The expected density factor of random minimizers has been shown to be 2 [13] and there is a general non-tight lower-bound of 1.5 [14].

3 Methods

The current heuristic algorithms that generate compact UHSs begin by constructing a minimum decycling set $\mathcal{D}_k$. We propose using $\mathcal{D}_k$ as an "approximate UHS" and defining an order based on it. The rationale for this idea is threefold. First, for most combinations of k and L, the majority of k-mers in UHSs generated by these heuristics belong to $\mathcal{D}_k$. Second, as discussed above, $\mathcal{D}_k$ can be generated very efficiently, while the subsequent k-mer additions that the heuristics perform in order to remove long paths are very slow. Third, Zheng et al. [24] showed that following the removal of a minimum decycling set from a complete dBG of order k, the longest remaining path has length $O(k^3)$, which bounds the length of the longest remaining sequence. Below we explore this idea and develop several variants of decycling set-based minimizer orders.

3.1 Decycling set-based minimizer orders

We define a partition-compatible order based only on a minimum decycling set $\mathcal{D}_k$ and use it in lieu of a UHS-based minimizer order. In this order, k-mers in $\mathcal{D}_k$ precede all other k-mers. Within each set, a random hash function is used to compare between k-mers. $\mathcal{D}_k$ can be constructed efficiently using Knuth's implementation of Mykkeltveit's algorithm [15] as described above.

For large values of k, when $\mathcal{D}_k$ is too large to store or takes too much time to compute, we can instead scan the target sequence and for every k-mer test its membership in $\mathcal{D}_k$ on the fly using the procedure outlined in Algorithm 1. The real parts of the embeddings of a k-mer x and its clockwise rotation x' are computed in $O(k)$ time and compared to determine if x is the first counter-clockwise rotation with $\mathbb{R}(x) > 0$. If $\mathbb{R}(x) = \mathbb{R}(x') = 0$, then the algorithm determines whether x is a lexicographically smallest rotation in $O(k)$ time.

Algorithm 1 Decycling set membership

```
Input: k-mer $x$, $|x| = k$
Output: Membership in the minimum decycling set $\mathcal{D}_k$
 1: for $i \in [0, k-1]$ do $c_i = \sin(2\pi i/k)$
 2: $\mathbb{R}(x) = \sum_{i=0}^{k-1} c_i x_i$
 3: $x' = x_{k-1}x_0x_1..x_{k-2}$
 4: $\mathbb{R}(x') = \sum_{i=0}^{k-1} c_i x'_i$
 5: if $\mathbb{R}(x) > 0$ then                ▷ Check if $x$ is the first rotation with $R(x) > 0$
 6:     if $\mathbb{R}(x') \leq 0$ then return true
 7: else if $\mathbb{R}(x) = 0$ then
 8:     if $\mathbb{R}(x') = 0$ then            ▷ Check if $x$ is the lexico. smallest rotation
 9:         $i \leftarrow 0$
10:         for $j \in [1, 2k-1]$ do
11:             if $x_{j \bmod k} < x_i$ then return false
12:             if $x_{j \bmod k} > x_i$ then $i = 0$
13:             else $i \leftarrow i + 1$
14:             if $(j \geq k-1) \wedge (i \bmod k = 0)$ then return true
15: return false
```

Proposition 1 (Alg. 1 correctness). *Alg. 1 correctly determines whether a k-mer is a member of $\mathcal{D}_k$ in time $O(k)$.*

Proof. The proof follows from the definition of $\mathcal{D}_k$. We say that a k-mer x is *positive, negative,* or *non-positive* if $\mathbb{R}(x) > 0, < 0$, or ≤ 0, respectively. Recall that a k-mer $x \in \mathcal{D}_k$ iff either: (i) it is the first positive counter-clockwise rotation in its conjugacy class; or, (ii) all k-mers in the conjugacy class have $\mathbb{R} = 0$ and x is a lexicographically smallest rotation.

For (i), line 6 returns true iff the input k-mer x is the first positive counter-clockwise rotation in its conjugacy class, i.e. x has $\mathbb{R}(x) > 0$ and the one letter clockwise rotation of x, denoted x' has $\mathbb{R}(x') \leq 0$.

For (ii), note that if two consecutive rotations of a k-mer x, x' have $\mathbb{R}(x) = \mathbb{R}(x') = 0$ (lines 7-8), then all k-mers in that conjugacy class have zero embedding (Lemma 1 in Mykkeltveit [15]). The loop in lines 10-14 checks all possible rotations of x and returns false if it finds a k-mer that is lexicographically smaller

than x (line 11), otherwise it returns true either if it checked all possible rotations and none of them is lexicographically smaller than x ($i = 0$ and $j \geq k-1$) or it finds that x is identical to one of its rotations and x is a lexicographically smallest rotation ($i = k$ and $j \geq k-1$).
The embedding computations (lines 1, 2, and 4) take $O(k)$ time. The loop beginning on line 10 can run for at most $2k$ times and performs constant time computations per iteration for a total running time of $O(k)$. □

3.2 Double decycling set-based minimizer orders

By symmetry, Mykkeltveit's construction can be used to create a minimum decycling set using the first counter-clockwise negative k-mer x in each conjugacy class rather than the first positive one. We refer to this set as the *symmetric* decycling set $\tilde{\mathcal{D}}_k$. The decycling set and symmetric decycling set divide sequences according to the following interesting property:

Theorem 1 (remaining path partition). *In any remaining path in the complete dBG after removing* $\mathcal{D}_k$*, all the positive nodes precede all the non-positive nodes.*

In other words, a remaining path must consist of two distinct parts: A *positive part*, containing only positive k-mers, followed by the second *non-positive part* consisting of non-positive k-mers only. The proof relies on two lemmas:

Lemma 1. *The k-mers associated with all incoming neighbours of a node x in a dBG have the same* $\mathbb{R}(x)$.

Proof. All incoming neighbours y of x differ only in y_0, and have embedding with $\mathbb{R}(y) = y_0 \sin(0) + \sum_{i=1}^{k-1} \sin(2\pi i/k) y_i = \sum_{i=1}^{k-1} \sin(2\pi i/k) y_i$.

Lemma 2. *The pure cycles factor the complete dBG, namely, every k-mer belongs to exactly one of the pure cycles.*

Proof. Every k-mer is on some pure cycle corresponding to its rotations. Assume the contrary that k-mer x is on two distinct pure cycles, C_1 and C_2. Let y be the last common node in the path in $C_1 \cap C_2$ starting from x. Then, the edges out of y in the two cycles are distinct, contradicting the fact that both correspond to the cyclic rotation of y.

Proof (Thm. 1). Let x_i be the first non-positive node in a remaining path $x_1, \ldots, x_t$ and assume the contrary that there exists a positive x_j for $j > i$. W.l.o.g. assume x_j is the first with that property in the path. Let C be the pure cycle that contains x_j. C exists and it is well defined by Lemma 2. Let y be the node preceding x_j in C. By Lemma 1, $\mathbb{R}(x_{j-1}) = \mathbb{R}(y)$. Since y is non-positive, x_j should be in $\mathcal{D}_k$ as the first positive node in C, a contradiction. □

By a similar argument, in a remaining path after removing $\tilde{\mathcal{D}}_k$, the negative nodes precede all other nodes. Thus, removal of a *double decycling set* consisting of $\mathcal{D}_k \cup \tilde{\mathcal{D}}_k$ would leave only short remaining paths that cannot contain both negative and positive k-mers.

We define a partition-compatible minimizer order based on double decycling sets with $\Pi = \{\mathcal{D}_k, \tilde{\mathcal{D}}_k \setminus \mathcal{D}_k, \Sigma^k \setminus (\mathcal{D}_k \cup \tilde{\mathcal{D}}_k)\}$. Because the double decycling set leaves even shorter remaining paths, we hypothesize that this minimizer order may achieve lower density compared to the one using only a single decycling set.

3.3 Modified decycling set-based minimizer orders

We defined another variant of the decycling set-based order to account for homopolymers. Long homopolymers in a sequence can increase the particular density of a decycling set-compatible minimizer order, and removing them from the set may improve the scheme performance. However, we note that all homopolymers have an embedding with $\mathbb{R}(x) = 0$. Since in practice k-mers with embedding 0 are only a small fraction of the decycling set, we simply choose to exclude all of them from the decycling set. The resulting set is denoted $\mathcal{D}'_k$, and we call the corresponding order *modified decycling set-based order*. This relieves us of the need to perform lines 7-14 in Algorithm 1, and as a result could speed up the membership test. Modified symmetric decycling sets $\tilde{\mathcal{D}}'_k$ are defined analogously, as is the modified double decycling set compatible order.

4 Results

We compared the performance of our new decycling set-based minimizer orders to UHS-based orders and random orders, across a range of k and L values. We measured performance using expected and particular density factors. Expected density factors were estimated by measuring density on five random i.i.d. sequences of 1M nt. Particular density factors were measured on a randomly selected 1M nt segment from chromosome X of the CHM13 telomere-to-telomere human genome assembly [16] with 10 repetitions using different seeds for the pseudo-random hash functions. We used Python's `hash` function as the pseudo-random hash to compare between k-mers within each set of a partition. Scripts to compute the expected and particular density of the different minimizer orders are available from github.com/OrensteinLab/DecyclingSetBasedMinimizerOrder.

4.1 Decycling set-based orders outperform UHS-based orders

Figure 2 compares the density factors of the tested orders for $k = 11$ and varying L values, and for $k = 5$ to 15 and $L = 100$. Average density factors over the repeated runs are shown for visual clarity. The same plots with error bars displayed are in Figure S1. The order denoted "decycling-UHS" is a variant of UHS order in which the decycling set k-mers precede the rest of the UHS. UHS- and decycling-UHS-compatible orders were generated by DOCKS for $k < 12$ and by PASHA for $12 \leq k \leq 13$. The sets of PASHA are slightly less compact than those of DOCKS, hence the slight bump in density factor for the UHS order at $k = 12$ (Figures 2A,B). UHSs for larger k could not be generated due to time- and storage-intensive computation required for every combination of k and L. In contrast, the decycling set-based orders have the distinct advantage of being easily computed on the fly for any (k, L) combination.

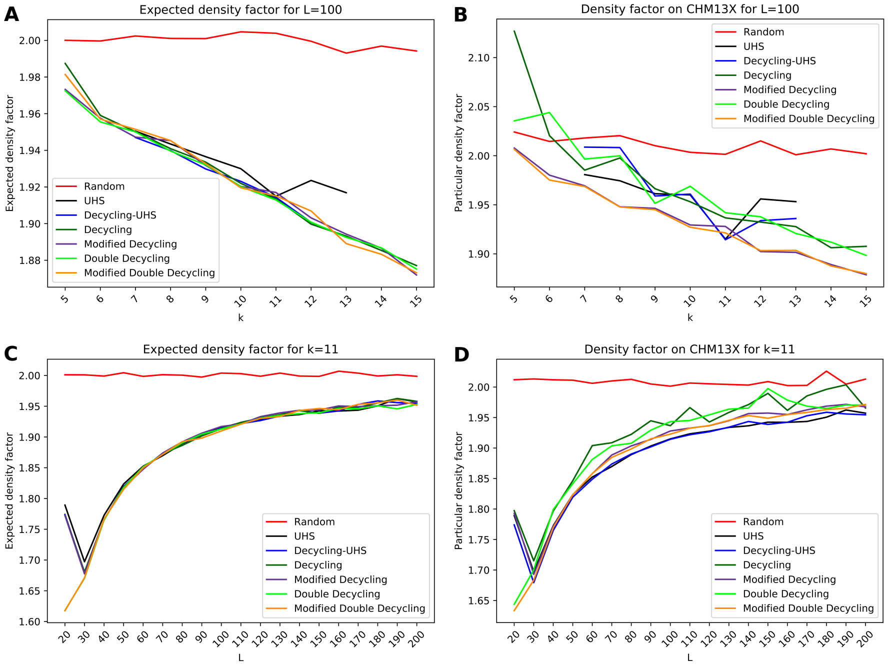

Fig. 2: **Density factor of decycling set-based minimizer orders and UHS-based orders.** The expected density (**A,C**) and particular density factors on CHM13X (**B,D**) of different minimizer orders is compared for fixed $L = 100$ and varying k (**A,B**) and fixed $k = 11$ and varying L (**C,D**).

The decycling set-based orders consistently perform similarly or better than UHS-based orders. As expected, random orders typically do worst, and the relative improvement of UHS- and decyling set-based orders compared to random orders increases with k. Conversely, as L grows for fixed k the density factors of the different methods are more similar. The particular density matches the expected density relatively closely for all orders but is much noisier (Figure S1).

4.2 Scaling to $k \geq 20$

We compared the decycling set-based orders to the random baseline orders for much larger k than is possible with UHS-compatible orders. Figure 3 shows

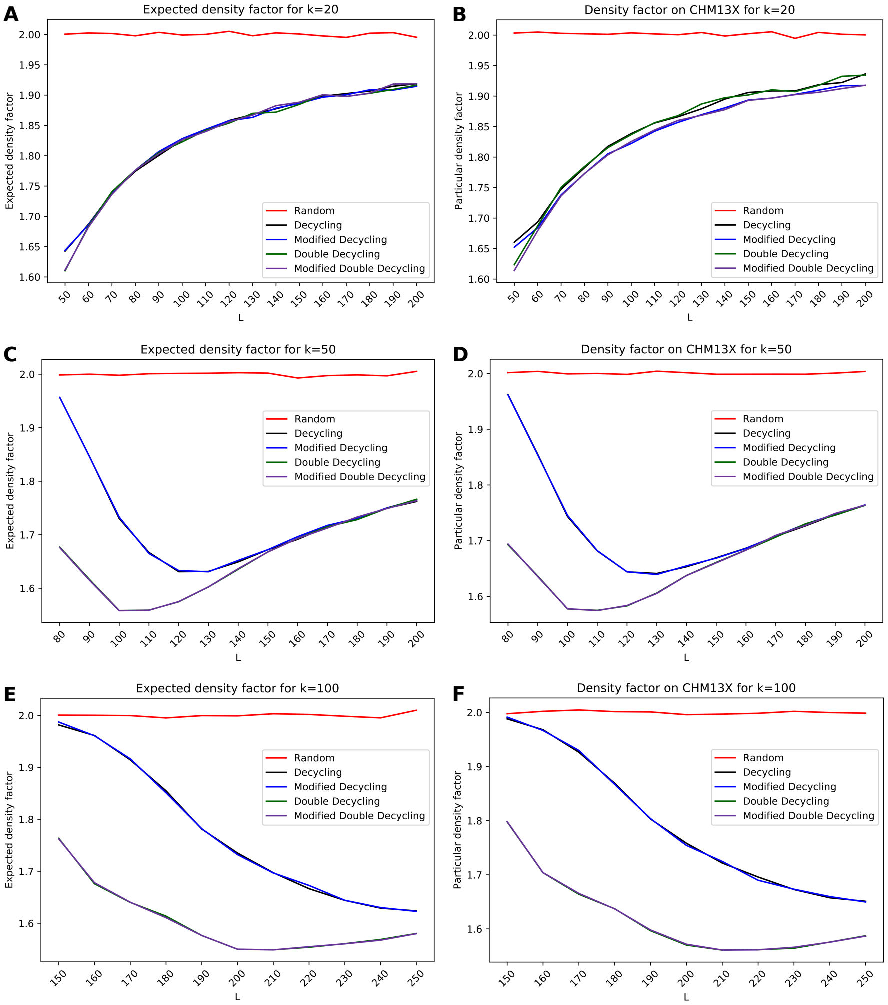

Fig. 3: **Density factors of decycling set-based orders for large k.** The expected density factor (**A,C,E**) and the particular density factor on CHM13X (**B,D,F**) of different minimizer orders is compared for fixed $k = 20$ (**A,B**), fixed $k = 50$ (**C,D**), and fixed $k = 100$ (**E,F**) for varying L. Note that in **C-F** the lines for the modified and unmodified orders are almost identical.

results for $k = 20, 50$ and 100. Average density factors over the repeated runs are shown. The same plots with error bars displayed are in Figure S2.

As k grows, the advantage of the decycling set-based order becomes even more pronounced and the double decycling set-based order improves more significantly over the decycling set-based order. This is true in particular for shorter L, with the differences between the decycling and double decycling set-based orders disappearing as L grows. At the same time, for larger k, the modified variants of the decycling set and double decycling set orders perform essentially the same as the original, but with improved k-mer query runtime. In all cases, the particular density factor is very close to that of the expected density factor.

5 Discussion

In this work, we solved one of the major limitations of UHS-based minimizer orders. By relieving the strict requirement of generating a set of k-mers that hits every L-long sequence, we were able to generate minimizer orders that are close to universal and can be calculated efficiently on the fly. Based on Mykkveltveit's algorithm, we developed a method to determine if a k-mer belongs to a minimum decycling set, which can be applied to any k. We demonstrated that minimizer orders based on minimum decycling sets are comparable or better in their density to minimizer orders based on UHSs, thus achieving good performance while avoiding escalating runtime and memory usage with the increase of k.

We also defined the modified and double decycling set orders. For longer k and relatively shorter L, the double decycling set-based order yields much lower density than even the decycling set-based order. Although we did not perform extensive runtime comparisons of the methods, the double decycling set-based order is generally slower to compute than the decycling set-based order, and the modified orders perform fewer computations and thus can be slightly faster. As the density of the different methods converges as L increases, this suggests using modified double decycling set-based order for smaller L to achieve lower density, while modified decycling set-based order can be used for larger L and achieve similar density with faster running times. Based on the results we have presented, a general rule-of-thumb appears to be that the advantage of the double decycling set persists until around $L = 2.5k$.

We see several promising future directions to take. First, it may be possible to more rigorously define which of the different decycling set-based orders is better to use for each given combination of k and L. Second, frequency-based orders are known to be highly efficient in terms of density while easily computable as sequence-specific minimizer orders. It will be interesting to extend our work by ranking each of the sets in a partition by their frequency in a specific sequence dataset to achieve lower density values (as was recently shown by incorporating UHS-based orders with frequency ranking [17]). Third, it may be possible to use decycling sets and their variants as sketches without defining compatible minimizer orders by simply including all decycling set k-mers in the sketch. By choosing an appropriate value of k and decycling set variant it may be possible to achieve a given desired density. Such schemes would be better conserved than minimizers as they are not dependent on a longer sequence window.

Our new approach can enable more efficient analyses of high-throughput DNA sequencing data. By implementing our new decycling set-based minimizer orders in data structures and algorithms of high-throughput DNA sequencing analysis, we expect to see reductions in runtime and memory usage, beyond what was previously demonstrated using UHS-based minimizer orders.

Acknowledgments

This study was supported by a United States–Israel Binational Science Foundation (BSF) grant no. 2020297 to YO and BB. RS was supported in part by the Israel Science Foundation (grant 2206/22) and by Len Blavatnik and the Blavatnik Family foundation. DP and LP were supported in part by fellowships from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University. LP was supported in part by the National Natural Science Foundation of China project 61902072.

References

1. Chikhi, R., Limasset, A., Medvedev, P.: Compacting de Bruijn graphs from sequencing data quickly and in low memory. Bioinformatics **32**(12), i201–i208 (2016)
2. Deorowicz, S., Kokot, M., Grabowski, S., Debudaj-Grabysz, A.: KMC 2: fast and resource-frugal k-mer counting. Bioinformatics **31**(10), 1569–1576 (2015), https://doi.org/10.1093/bioinformatics/btv022
3. Edgar, R.: Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. PeerJ **9**, e10805 (2021)
4. 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 (2021)
5. Ekim, B., Berger, B., Orenstein, Y.: A randomized parallel algorithm for efficiently finding near-optimal universal hitting sets. In: Research in Computational Molecular Biology. pp. 37–53. Springer International Publishing (2020)
6. Flomin, D., Pellow, D., Shamir, R.: Data set-adaptive minimizer order reduces memory usage in k-mer counting. Journal of Computational Biology (2022)
7. Fredricksen, H., Maiorana, J.: Necklaces of beads in k colors and k-ary de bruijn sequences. Discrete Mathematics **23**(3), 207–210 (1978), https://www.sciencedirect.com/science/article/pii/0012365X7890002X
8. Hoang, M., Zheng, H., Kingsford, C.: Differentiable learning of sequence-specific minimizer schemes with deepminimizer. Journal of Computational Biology (2022)
9. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
10. Knuth, D.E.: Unavoidable2. http://www-cs-faculty.stanford.edu/~uno/programs/unavoidable2.w (2003)
11. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics **34**(18), 3094–3100 (2018)
12. Li, Y., Kamousi, P., Han, F., Yang, S., Yan, X., Suri, S.: Memory efficient minimum substring partitioning. In: Proceedings of the VLDB Endowment. vol. 6, pp. 169–180. VLDB Endowment (2013)
13. Marçais, G., Pellow, D., Bork, D., Orenstein, Y., Shamir, R., Kingsford, C.: Improving the performance of minimizers and winnowing schemes. Bioinformatics **33**(14), i110–i117 (2017), https://doi.org/10.1093/bioinformatics/btx235
14. Marçais, G., DeBlasio, D., Kingsford, C.: Asymptotically optimal minimizers schemes. Bioinformatics **34**(13), i13–i22 (2018), https://doi.org/10.1093/bioinformatics/bty258
15. Mykkeltveit, J.: A proof of Golomb's conjecture for the de Bruijn graph. Journal of Combinatorial Theory, Series B **13**(1), 40–45 (1972), http://www.sciencedirect.com/science/article/pii/0095895672900068

16. Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A.V., Mikheenko, A., Vollger, M.R., Altemose, N., Uralsky, L., Phillippy, A.M., et al.: The complete sequence of a human genome. Science **376**(6588), 44–53 (2022), https://www.science.org/doi/abs/10.1126/science.abj6987
17. Nyström-Persson, J., Keeble-Gagnère, G., Zawad, N.: Compact and evenly distributed k-mer binning for genomic sequences. Bioinformatics **37**(17), 2563–2569 (2021)
18. 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**(10), e1005777 (2017)
19. Pellow, D., Dutta, A., Shamir, R.: Parameterized syncmer schemes improve long-read mapping. bioRxiv (2022)
20. Rautiainen, M., Marschall, T.: MBG: Minimizer-based sparse de Bruijn Graph construction. Bioinformatics **37**(16), 2476–2478 (2021), https://doi.org/10.1093/bioinformatics/btab004
21. Ruskey, F., Savage, C., Wang, T.M.Y.: Generating necklaces. Journal of Algorithms **13**(3), 414–430 (1992)
22. Sahlin, K.: Flexible seed size enables ultra-fast and accurate read alignment. bioRxiv (2022), https://www.biorxiv.org/content/early/2022/05/25/2021.06.18.449070
23. 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. pp. 76–85 (2003)
24. Zheng, H., Kingsford, C., Marçais, G.: Lower density selection schemes via small universal hitting sets with short remaining path length. In: Research in Computational Molecular Biology. pp. 202–217. Springer International Publishing (2020)