Theoretical Computer Science 1076 (2026) 115932

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Minimizing the minimizers via alphabet reordering

Hilde Verbeek [a,1], Lorraine A.K. Ayad [b], Grigorios Loukides [c,*], Solon P. Pissis [a,d,2]

[a] Centrum Wiskunde & Informatica, Science Park 123, Amsterdam, 1098 XG, The Netherlands
[b] Brunel University of London, Kingston Lane, Uxbridge, Middlesex, UB8 3PH, United Kingdom
[c] King's College London, Strand Campus, 30 Aldwych, London, WC2B 4BG, United Kingdom
[d] Vrije Universiteit, De Boelelaan 1105, Amsterdam, 1081 HV, The Netherlands

ARTICLE INFO

Section Editor: Seth Pettie
Handling Editor: Katie Harris

Keywords:
Minimizers
Sampling
Alphabet reordering
Feedback arc set
Eulerian feedback arc set

ABSTRACT

Minimizer sampling is one of the most widely-used mechanisms for sampling strings [Schleimer et al., SIGMOD 2003; Roberts et al., Bioinformatics 2004]. Let $S = S[1] \dots S[n]$ be a string over a totally ordered alphabet Σ. Further let $w \geq 2$ and $k \geq 1$ be two integers. The minimizer of $S[i \,..\, i + w + k - 2]$ is the smallest position in $[i, i + w - 1]$ where the lexicographically smallest length-k substring of $S[i \,..\, i + w + k - 2]$ starts. The set of minimizers over all $i \in [1, n - w - k + 2]$ is the set $\mathcal{M}_{w,k}(S)$ of the minimizers of S. We consider the following basic problem: *Given S, w, and k, can we efficiently compute a total order on Σ that minimizes $|\mathcal{M}_{w,k}(S)|$?* We show that this is unlikely by proving that the problem is NP-hard *for any* $w \geq 2$ *and* $k \geq 1$. Our result provides theoretical justification as to why *there exist no exact algorithms* for minimizing the minimizer samples, while *there exists a plethora of heuristics* for the same purpose.

1. Introduction

The minimizer sampling mechanism has been introduced independently by Schleimer et al. [1] and by Roberts et al. [2]. Since its inception, it has been employed ubiquitously in modern sequence analysis methods underlying some of the most widely-used tools [3–5].

Let $S = S[1] \dots S[n]$ be a string over a totally ordered alphabet Σ. Further let $w \geq 2$ and $k \geq 1$ be two integers. The minimizer of the fragment $S[i \,..\, i + w + k - 2]$ of S is the smallest position in $[i, i + w - 1]$ where the lexicographically smallest length-k substring of $S[i \,..\, i + w + k - 2]$ starts. We then define the set $\mathcal{M}_{w,k}(S)$ of the minimizers of S as the set of the minimizers positions over all fragments $S[i \,..\, i + w + k - 2]$, for $i \in [1, n - w - k + 2]$. Every fragment $S[i \,..\, i + w + k - 2]$ containing w length-k fragments is called a *window* of S.

Example 1. Let $S = \texttt{aacaaacgcta}$, $w = 3$, and $k = 3$. Assuming $\texttt{a} < \texttt{c} < \texttt{g} < \texttt{t}$, we have that $\mathcal{M}_{w,k}(S) = \{1, 4, 5, 6, 7\}$. The minimizers positions are colored red: $S = \texttt{aacaaacgcta}$.

* Corresponding author.
E-mail addresses: hilde.verbeek@cwi.nl (H. Verbeek), lorraine.ayad@brunel.ac.uk (L.A.K. Ayad), grigorios.loukides@kcl.ac.uk (G. Loukides), solon.pissis@cwi.nl (S.P. Pissis).

[1] Supported by CWI's Constance van Eeden Fellowship.
[2] Partially supported by the PANGAIA and ALPACA projects that have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No. 872539 and 956229, respectively.

https://doi.org/10.1016/j.tcs.2026.115932
Received 5 August 2025; Received in revised form 15 February 2026; Accepted 26 March 2026
Available online 3 April 2026

Note that by choosing the *smallest* position in $[i, i+w-1]$ where the lexicographically smallest length-k substring starts, we resolve ties in case the latter substring has multiple occurrences in a window.

It is easy to prove that minimizer samples enjoy the following three useful properties [6]:

- **Property 1 (approximately uniform sampling)**: Every fragment of length at least $w+k-1$ of S has at least one representative position sampled by the mechanism.
- **Property 2 (local consistency)**: Exact matches between fragments of length at least $\ell \geq w+k-1$ of S are preserved by means of having the same (relative) representative positions sampled by the mechanism.
- **Property 3 (left-to-right parsing)**: The minimizer selected by any fragment of length $w+k-1$ comes at or after the minimizers positions selected by all previous windows.

Since Properties 1 to 3 hold *unconditionally*, and since the ordering of letters does not affect the correctness of algorithms using minimizer samples [7–10], one would like to choose the ordering that minimizes the resulting sample as a means to improve the space occupied by the underlying data structures; contrast Example 1 to the following example.

Example 2. Let $S = \texttt{aacaaacgcta}$, $w = 3$, and $k = 3$. Assuming $\texttt{c} < \texttt{a} < \texttt{g} < \texttt{t}$, we have that $\mathcal{M}_{w,k}(S) = \{3, 6, 7\}$. The minimizers positions are colored red: $S = \texttt{aacaaacgcta}$. In fact, this ordering is a best solution in minimizing $|\mathcal{M}_{w,k}(S)|$, together with the orderings $\texttt{c} < \texttt{g} < \texttt{t} < \texttt{a}$ and $\texttt{c} < \texttt{g} < \texttt{a} < \texttt{t}$, which both, as well, result in $|\mathcal{M}_{w,k}(S)| = 3$.

Our problem. We next formalize the problem of computing a best such total order on Σ:

MINIMIZING THE MINIMIZERS
Input: A string $S \in \Sigma^n$ and two integers $w \geq 2$ and $k \geq 1$.
Output: A total order on Σ that minimizes $|\mathcal{M}_{w,k}(S)|$.

Motivation. A lot of effort has been devoted by the bioinformatics community to designing practical algorithms for minimizing the resulting minimizer sample [11–17]. Most of these approaches consider the space of all orderings on Σ^k (the set of all possible length-k strings on Σ) instead of the ones on Σ; and employ *heuristics* to choose some ordering resulting in a small sample (see Section 4 for a discussion). To illustrate the impact of reordering on the number of minimizers, we considered two real-world datasets. We investigated a *subset* of the possible orderings and found a large gap between the worst (largest number of minimizers) and the best (smallest number of minimizers) reordering within the subset, despite the fact that we have not explored the entire space of orderings. The first dataset we considered is the complete genome of Escherichia coli str. K-12 substr. MG1655. For selecting minimizers, we considered different orderings on Σ^k, for variable k. To do this, we mapped every length-k substring to its lexicographic rank in $\{\texttt{A},\texttt{C},\texttt{G},\texttt{T}\}^k$ (assuming $\texttt{A} < \texttt{C} < \texttt{G} < \texttt{T}$) constructing a new string S over $[1, |\Sigma|^k]$. We then computed $|\mathcal{M}_{w,1}(S)|$ for different values of w and k and orderings on $[1, |\Sigma|^k]$, as individual letters in the new string correspond one-to-one to length-k substrings in the original string. It should be clear that this process corresponds to computing the size of $\mathcal{M}_{w,k}$ for the original sequence over $\{\texttt{A},\texttt{C},\texttt{G},\texttt{T}\}$ allowing any arbitrary orderings of length-k substrings rather than just lexicographical orderings. The second dataset is the complete genome of SARS-CoV-2 OL663976.1. Fig. 1 shows the min and max values of the size of the obtained minimizer samples. The results in Fig. 1 clearly show the impact of alphabet reordering on $|\mathcal{M}_{w,1}(S)|$: the gap between the min and max is quite significant as in all cases we have $3\,\text{min} < \text{max}$. See Section 5 for more experimental results.

This begs the question:

Given S, w, and k, can we efficiently compute a total order on Σ that minimizes $|\mathcal{M}_{w,k}(S)|$?

Our contribution. We answer this basic question in the negative. Let us first define the decision version of MINIMIZING THE MINIMIZERS.

MINIMIZING THE MINIMIZERS (DECISION)
Input: A string $S \in \Sigma^n$ and three integers $w \geq 2$, $k \geq 1$, and $\ell > 0$.
Output: Is there a total order on Σ such that $|\mathcal{M}_{w,k}(S)| \leq \ell$?

Our main contributions are the following two lemmas implying Theorem 1.

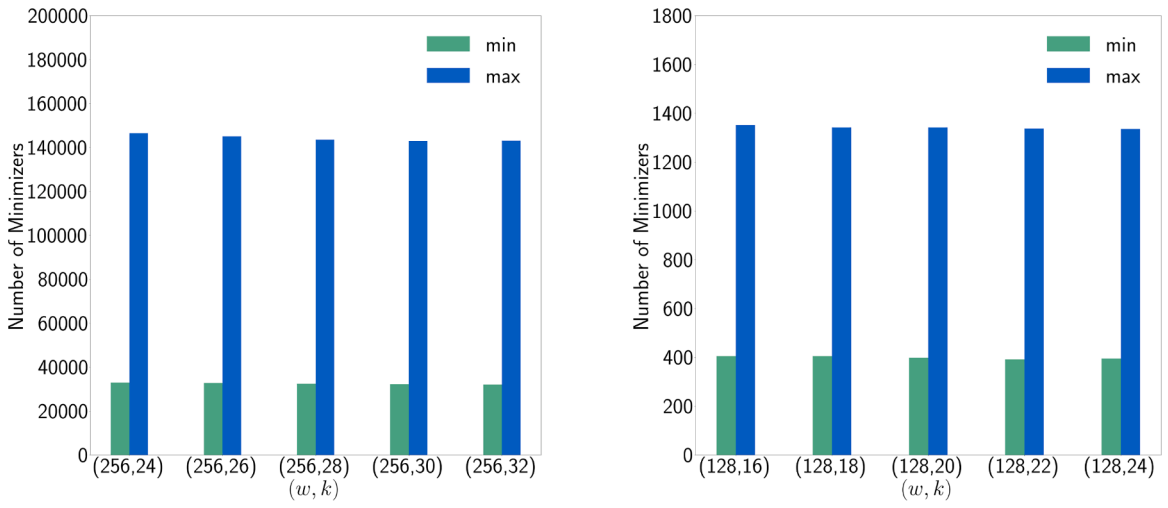
Lemma 1. *MINIMIZING THE MINIMIZERS (DECISION) is NP-complete if $w \geq 3$ and $k \geq 1$.*

Lemma 2. *MINIMIZING THE MINIMIZERS (DECISION) is NP-complete if $w = 2$ and $k \geq 1$.*

Theorem 1 (Main Result)**.** *MINIMIZING THE MINIMIZERS (DECISION) is NP-complete if $w \geq 2$.*

Theorem 1 provides theoretical justification as to why *there exist no exact algorithms* for minimizing the minimizer samples, while *there exists a plethora of heuristics* for the same purpose. Notably, Theorem 1 settles the complexity landscape of the MINIMIZING THE MINIMIZERS problem. To cover *all combinations* of input parameters w and k (i.e., for any $w \geq 2$ and $k \geq 1$), we design two non-trivial reductions from the feedback arc set problem [18]. In particular, the reduction in Lemma 1 ($w \geq 3$) starts with any directed graph, whereas the reduction in Lemma 2 ($w = 2$) starts with any Eulerian directed graph.

The reductions we present are specifically for the case in which the size of the alphabet Σ is variable. If $|\Sigma|$ is bounded by a constant, the problem can be solved in polynomial time: one can enumerate the $|\Sigma|!$ permutations of the alphabet, compute the number of minimizers for each ordering in linear time [19], and output a globally best ordering.



(a) Complete genome of Escherichia coli. (b) Complete genome of SARS-CoV-2.

Fig. 1. The min and max values of the size of the minimizer sample, among *some* of the possible orderings of $[1, |\Sigma|^k]$, on two real datasets using a range of (w, k) parameter values. Although $|\Sigma| = 4$, we construct a string S over $[1, |\Sigma|^k]$ and compute $|\mathcal{M}_{w,1}(S)|$.

Other related work. Choosing a best total order on Σ is generally not new; it has also been investigated in other contexts, e.g., for choosing a best total order for minimizing the number of runs in the Burrows-Wheeler transform [20]; for choosing a best total order for minimizing (or maximizing) the number of factors in a Lyndon factorization [21]; or for choosing a best total order for minimizing the number of bidirectional string anchors [9].

Paper organization. Section 2 presents the proof of Lemma 1 ($w \geq 3$). Section 3 presents the proof of Lemma 2 ($w = 2$). Section 4 presents a discussion on orderings on Σ^k in light of Theorem 1. In Section 5, we present further experimental results demonstrating the impact of alphabet reordering on other, more involved, minimizer schemes. We conclude this paper with a few open questions in Section 6.

A preliminary version of this paper *without Lemma 2 and Section 5* was announced at CPM 2024 [22].

2. Minimizing the Minimizers is NP-complete for $w \geq 3$

We show that the Minimizing the Minimizers problem is NP-hard by a reduction from the well-known Feedback Arc Set problem [18]. Let us first formally define the latter problem.

Feedback Arc Set
Input: A directed graph $G = (V, A)$.
Output: A set $F \subseteq A$ of minimum size such that $(V, A \setminus F)$ contains no directed cycles.

We call any such $F \subseteq A$ a *feedback arc set*. The decision version of the Feedback Arc Set problem is naturally defined as follows.

Feedback Arc Set (Decision)
Input: A directed graph $G = (V, A)$ and an integer $\ell' > 0$.
Output: Is there a set $F \subseteq A$ such that $(V, A \setminus F)$ contains no directed cycles and $|F| \leq \ell'$?

An equivalent way of phrasing this problem is to find an ordering on the set V of the graph's vertices, such that the number of arcs (u, v) with $u > v$ is minimized [23]. Then this is a topological ordering of the graph $(V, A \setminus F)$, and will be analogous to the alphabet ordering in the Minimizing the Minimizers problem when the vertex set V is the alphabet Σ; see [9] for a similar application of this idea.[2] If Minimizing the Minimizers is then solved on the instance constructed by our reduction, producing a total order on

[2] Our proof is more general and thus involved because it works for any values $w \geq 3$ and $k \geq 1$, whereas the reduction from [9] works only for some fixed parameter values.

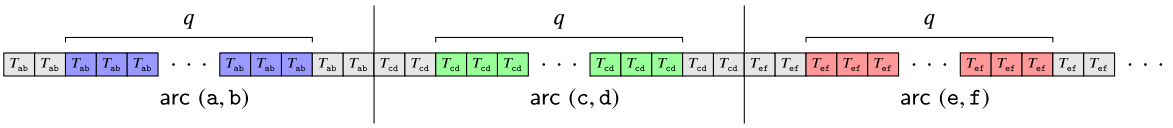


Fig. 2. Illustration of the structure of string S, with the different gadgets for different arcs in G. The highlighted blocks are the ones for which the minimizers are counted in M_G and M_B.

V, taking all arcs (u, v) with $u > v$ should produce a feedback arc set of minimum size, solving the original instance of the FEEDBACK ARC SET problem.

2.1. Overview of the technique

Given any instance $G = (V, A)$ of FEEDBACK ARC SET, we will construct a string S over alphabet $\Sigma = V$ and of length polynomial in $|A|$. Specifically, we define string S as follows:

$$S = \prod_{(\mathtt{a},\mathtt{b})\in A} T_{\mathtt{ab}}^{q+4},$$

where $T_{\mathtt{ab}}$ is a string consisting of the letters $\mathtt{a}$ and $\mathtt{b}$, whose length depends only on w and k, and q is an integer polynomial in $|A|$, both of which will be defined later. The product $\prod$ of some strings is defined as their concatenation, and X^q denotes q concatenations of string X starting with the empty string; e.g., if $X = \mathtt{ab}$ and $q = 4$, we have $X^q = (\mathtt{ab})^4 = \mathtt{abababab}$.

String $T_{\mathtt{ab}}$ will be designed such that each occurrence, referred to as a *block*, will contain few minimizers if $\mathtt{a} < \mathtt{b}$ in the alphabet ordering, and many minimizers if $\mathtt{b} < \mathtt{a}$, analogous to the "penalty" of removing the arc $(\mathtt{a}, \mathtt{b})$ as part of the feedback arc set. For any occurrence of $T_{\mathtt{ab}}$ in S, provided that it is both preceded and followed by two more occurrences of $T_{\mathtt{ab}}$, we count the number of minimizers starting within the current block, conditionally depending on the alphabet ordering. We denote by M_G the number of such minimizers when $\mathtt{a} < \mathtt{b}$, and by M_B the number when $\mathtt{b} < \mathtt{a}$ (the letters G and B stand for "good" and "bad", respectively). Note that M_G and M_B are constants depending only on how $T_{\mathtt{ab}}$ is constructed: the number of minimizers starting in these blocks equals one of the two values, regardless of the two letters of which the block is constituted of. This will allow us (see Fig. 2) to express the total number of minimizers in S in terms of $|F|$, the size of the feedback arc set, and some small *discrepancy term* that we denote by λ. We aim to create a string in which the "bad" blocks are contributed by arcs in the feedback arc set F, and the "good" blocks by arcs that are not, meaning we get the following formula for $\mathcal{M}_{w,k}(S, F)$:

$$\begin{aligned}\mathcal{M}_{w,k}(S, F) &= q \cdot M_B \cdot |F| + q \cdot M_G \cdot (|A| - |F|) + \lambda \\ &= q \cdot (M_B - M_G) \cdot |F| + q \cdot M_G \cdot |A| + \lambda. \end{aligned} \tag{1}$$

The discrepancy term λ is contributed by the $4 \cdot |A|$ blocks $T_{\mathtt{ab}}$ that are not preceded or followed by two occurrences of $T_{\mathtt{ab}}$ itself; namely, those that occur near some $T_{\mathtt{cd}}$, for another arc $(\mathtt{c}, \mathtt{d})$, or those that occur near the start or the end of S. The number of minimizers starting in such blocks may differ from M_G or M_B; however, we can determine upper and lower bounds on the total number of uncounted minimizers:

Lemma 3. *$|A| - 1 \le \lambda \le 4 \cdot |A| \cdot |T_{ab}|$ if $|T_{ab}| \ge \frac{1}{4}(w + k - 1)$.*

Proof. We are counting the number of minimizers in q blocks of $T_{\mathtt{ab}}$, for each arc $(\mathtt{a}, \mathtt{b})$. Note that we ignore four blocks for each arc, which is $4 \cdot |A|$ blocks of length $|T_{\mathtt{ab}}|$ each. This is $4 \cdot |A| \cdot |T_{\mathtt{ab}}|$ positions in total, which gives the upper bound on the number of disregarded minimizers. For the lower bound, note that, by hypothesis, four consecutive blocks are at least as long as a single minimizer window, meaning at least one minimizer must be missed among the four blocks surrounding the border between each pair of consecutive arcs. The lower bound follows by the fact that for $|A|$ arcs we have $|A| - 1$ such borders. □

With the equation for $\mathcal{M}_{w,k}(S, F)$ and the bounds on λ in mind, we can prove the following relationship between $\mathcal{M}_{w,k}(S, F)$ and $|F|$:

Lemma 4. *Let ℓ' be some positive integer and let $\ell = q \cdot (M_B - M_G) \cdot (\ell' + 1) + q \cdot M_G \cdot |A|$. Then $\mathcal{M}_{w,k}(S, F) \le \ell$ if and only if $|F| \le \ell'$, provided that*

- *$M_B > M_G$,*
- *$|T_{ab}| \ge \frac{1}{4}(w + k - 1)$, and*
- *q is chosen such that $\lambda < q \cdot (M_B - M_G)$.*

Proof. By hypothesis, $M_B - M_G$ is positive, thus, by Eq. (1), $\mathcal{M}_{w,k}(S, F)$ grows linearly with $|F|$. Suppose we have a feedback arc set F with $|F| \le \ell'$. Consider the alphabet ordering inducing F and let λ be the corresponding discrepancy for $\mathcal{M}_{w,k}(S, F)$. By hypothesis, we have $\lambda < q \cdot (M_B - M_G)$. Substituting the bounds on $|F|$ and λ into Eq. (1) gives

$$\begin{aligned}\mathcal{M}_{w,k}(S, F) &\le q \cdot (M_B - M_G) \cdot \ell' + q \cdot M_G \cdot |A| + q \cdot (M_B - M_G) \\ &= q \cdot (M_B - M_G) \cdot (\ell' + 1) + q \cdot M_G \cdot |A| = \ell, \end{aligned}$$

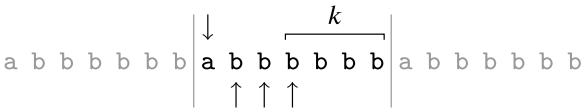


Fig. 3. Illustration of 3 copies of $T_{\mathtt{ab}}$ in S for $w = 7$ and $k = 4$ (Case A), along with its respective minimizers when $\mathtt{a} < \mathtt{b}$ (top) and when $\mathtt{b} < \mathtt{a}$ (bottom). It can be seen that $M_G = 1$ and $M_B = 3$.

completing the proof in one direction.

For the other direction, suppose we have picked F such that $\mathcal{M}_{w,k}(S, F) \leq \ell$ and assume that $|F| \geq \ell' + 1$ towards a contradiction. Then we have the following two inequalities:

$$\mathcal{M}_{w,k}(S, F) \leq \ell = q \cdot (M_B - M_G) \cdot (\ell' + 1) + q \cdot M_G \cdot |A|$$
$$\mathcal{M}_{w,k}(S, F) \geq q \cdot (M_B - M_G) \cdot (\ell' + 1) + q \cdot M_G \cdot |A| + \lambda. \quad \text{(by Eq. (1))}$$

By Lemma 3, for any non-trivial instance with $|A| > 1$, λ is strictly positive, meaning these inequalities are contradictory. Therefore, if $\mathcal{M}_{w,k}(S, F) \leq \ell$, it must be that $|F| \leq \ell'$. □

Given w and k, we must determine a string $T_{\mathtt{ab}}$ such that $M_B > M_G$ and $|T_{\mathtt{ab}}| \geq \frac{1}{4}(w + k - 1)$. We then simply have to choose some q, which is polynomial in $|A|$, satisfying $\lambda < q \cdot (M_B - M_G)$. At that point we will have constructed a string S for which it holds that the feedback arc set induced by the minimum set of minimizers is also a minimum feedback arc set on G, thus completing the reduction.

The following three subsections address the $T_{\mathtt{ab}}$ construction:

- Section 2.2: $w \geq k + 2$ (Case A);
- Section 2.3: $w = 3$ and $k \geq 2$ (Case B);
- Section 2.4: $3 < w < k + 2$ (Case C).

It should be clear that the above sections cover all the cases for $w \geq 3$ and $k \geq 1$. Section 2.5 puts everything together to complete the proof of Lemma 1.

2.2. Case A: $w \geq k + 2$

Lemma 5. *Let $T_{ab} = ab^{w-1}$, for $w \geq k + 2$. Then $M_G = 1$ and $M_B = w - k$.*

Proof. The block has length w; inspect Fig. 3. Recall that, for the window starting at position i, the candidates for its minimizer are the length-k fragments starting at positions $[i, i + w - 1]$. Therefore, for every window starting in a block $T_{\mathtt{ab}}$ (provided it is succeeded by another $T_{\mathtt{ab}}$), a candidate minimizer is $\mathtt{ab}^{k-1}$; so if $\mathtt{a} < \mathtt{b}$, each $T_{\mathtt{ab}}$ will contain just one minimizer. Thus we have $M_G = 1$.

For $\mathtt{b} < \mathtt{a}$, consider that $T_{\mathtt{ab}}$ contains $w - k$ occurrences of $\mathtt{b}^k$, and that for each window, at least one of the candidates for its minimizer is $\mathtt{b}^k$. Since there is no length-k substring that is lexicographically smaller than $\mathtt{b}^k$, each occurrence of $\mathtt{b}^k$ (and nothing else) is a minimizer, so it follows that $M_B = w - k$. Note that $M_B > M_G$ only if $w \geq k + 2$. □

2.3. Case B: $w = 3$ and $k \geq 2$

Lemma 6. *Let $T_{ab} = (ab)^t bb$ with $t = \left\lceil \frac{w+k}{2} \right\rceil$, for $w = 3$ and $k \geq 2$. Then $M_G = \left\lfloor \frac{k}{2} \right\rfloor + 3$ and $M_B = \left\lfloor \frac{k}{2} \right\rfloor + 4$.*

Proof. Since $w = 3$, for every window, the minimizer is one out of three length-k fragments; inspect Fig. 4. Every $\mathtt{a}$ in the block has a $\mathtt{b}$ before it. For any window starting at a position preceding an $\mathtt{a}$, two of the candidates start with a $\mathtt{b}$ and the other starts with an $\mathtt{a}$. As an example consider the window $\mathtt{babab}$ preceding an $\mathtt{a}$ in Fig. 4. We have that the first and the third candidates start with a $\mathtt{b}$ and the second starts with an $\mathtt{a}$. Therefore, if $\mathtt{a} < \mathtt{b}$, the candidate starting with an $\mathtt{a}$ will be chosen and every $\mathtt{a}$ in $T_{\mathtt{ab}}$ is a minimizer. Only the window starting at the third-to-last position of the block will not consider any length-k substring starting with an $\mathtt{a}$ as its minimizer, as therein we have three $\mathtt{b}$'s occurring in a row. Since $k \geq 2$, the last $\mathtt{b}$ of the block will be chosen if $\mathtt{a} < \mathtt{b}$. Thus, M_G counts every $\mathtt{a}$ and one $\mathtt{b}$, which gives:

$$M_G = t + 1 = \left\lceil \frac{w + k}{2} \right\rceil + 1 = \left\lceil \frac{3 + k}{2} \right\rceil + 1 = \left\lfloor \frac{k + 2}{2} \right\rfloor + 1 + 1 = \left\lfloor \frac{k}{2} \right\rfloor + 3.$$

For M_B, we apply the same logic to conclude that every $\mathtt{b}$ surrounded by $\mathtt{a}$'s is a minimizer, which accounts for all $\mathtt{b}$'s except the final three occurring at positions $[2t, 2t + 2]$. We will now prove that all three of these final $\mathtt{b}$'s are minimizers:

- For the window starting at position $2t$, the three minimizer candidates start, respectively, with $\mathtt{bb}$, $\mathtt{bb}$ and $\mathtt{ba}$. Since $k \geq 2$ and $\mathtt{b} < \mathtt{a}$, the first of these three (at position $2t$) will be the minimizer.
- For the window starting at position $2t + 1$, the first two candidates start, respectively, with $\mathtt{bb}$ and $\mathtt{ba}$, and the third starts with an $\mathtt{a}$. Again, the first candidate ($2t + 1$) will be the minimizer.

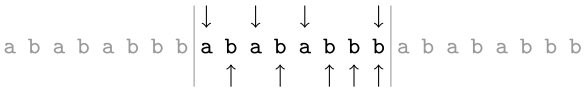


Fig. 4. $T_{\mathtt{ab}}$ for $w = 3$ and $k = 3$ (Case B), with its respective minimizers. The last b is a minimizer even when a < b, because $w = 3$. In this situation, $M_G = 4$ and $M_B = 5$.

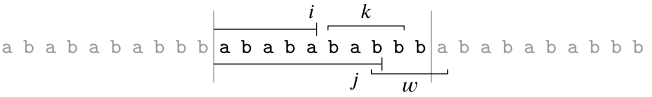


Fig. 5. $T_{\mathtt{ab}}$ for $w = 4$ and $k = 4$ (Case C), showing the positions i and j for counting M_G. Position i is the final position at which the length-k fragment does not contain bb, whereas j is the final position for which the starting position of the next $T_{\mathtt{ab}}$-block is not a candidate. When a < b, the minimizers in the block are all a's up to position $\max\{i, j+1\}$.

- For the window starting at position $2t+2$, the first and third candidates start with a b whereas the second starts with an a. The third candidate starts at the second position of the next $T_{\mathtt{ab}}$-block. Since $2t > k+1$, this candidate consists of only baba… alternating for k letters. It is equal to the first candidate, so by tie-breaking the first candidate $(2t+2)$ is the minimizer.

Thus, every b in the block will be a minimizer if b < a, and we have:

$$M_B = t + 2 = \left\lceil \frac{3+k}{2} \right\rceil + 2 = \left\lfloor \frac{k}{2} \right\rfloor + 4.$$

□

2.4. Case C: $3 < w < k+2$

Lemma 7. *Let $T_{ab} = (ab)^t bb$ with $t = \left\lceil \frac{w+k}{2} \right\rceil$, for $3 < w < k+2$. Then*

- *if k is even, $M_G = \frac{k}{2} + 2 + p$ and $M_B = \frac{k}{2} + 3 + p$, where $p = (w+k) \bmod 2$;*
- *if k is odd, $M_G = \left\lfloor \frac{k}{2} \right\rfloor + 3$ and $M_B = \left\lfloor \frac{k}{2} \right\rfloor + 4$.*

Proof. Every length-w fragment of the block contains at least one a and at least one b; inspect Fig. 5. Because of this, only a's will be minimizers if a < b and only b's if b < a (unlike when $w = 3$, as shown in Section 2.3). We start by counting M_G. Suppose we are determining the minimizer at position i. Every candidate we consider is a string of alternating a's and b's (starting with an a), in which potentially one a is substituted by a b (if the length-k fragment contains the bbb at the end of the block). A lexicographically smallest length-k fragment is one in which this extra b appears the latest, or not at all.

First, we will consider the number of length-k fragments in which the extra b does not occur. For these fragments, it is the case that no other fragment in the block is lexicographically smaller when a < b, so it is automatically picked as minimizer at the position corresponding to the start of the length-k fragment. The extra b appears at position $2t+1$ in the block, so this applies to all length-k fragments starting with an a that end before position $2t+1$. That is, all a's up to (and including) position $i = 2t - (k-1) = 2\left\lceil \frac{w+k}{2} \right\rceil - k + 1 = w + k + p - k + 1 = w + p + 1$, where $p = (w+k) \bmod 2$.

Next, we consider the length-k fragments that do include the extra b. At any position past i, the smallest candidate will be the first one starting with an a, *unless* one of the candidates appears in the next $T_{\mathtt{ab}}$-block, in which case the minimizer will be the first position of this next block (because this candidate does not include the extra b and is therefore smaller than any candidate before it). Specifically, this is the case if position $|T_{\mathtt{ab}}| + 1$ is one of the w candidates. Therefore, all windows starting at positions up to and including $j = |T_{\mathtt{ab}}| + 1 - w = \left(2\lceil \frac{w+k}{2} \rceil + 2\right) + 1 - w = w + k + 3 + p - w = k + p + 3$ will have as their minimizer the first position with an a, meaning that all a's up to position $j+1$ are minimizers.

We now have that all a's up to position $i = w + p + 1$ and all a's up to position $j + 1 = k + p + 4$ are minimizers. Thus we need to count the a's up to position $\max\{w + p + 1, k + p + 4\}$. Because, by hypothesis, $w < k + 2$, this maximum is equal to $k + p + 4$. The first $k + p + 4$ letters of the block are alternating a's and b's, so we get

$$M_G = \left\lceil \frac{k+p+4}{2} \right\rceil = \left\lceil \frac{k+p}{2} \right\rceil + 2 = \begin{cases} \frac{k}{2} + 2 + p & \text{if } k \text{ is even;} \\ \left\lfloor \frac{k}{2} \right\rfloor + 3 & \text{if } k \text{ is odd.} \end{cases}$$

Next, we compute M_B. We start by showing that the final three b's in $T_{\mathtt{ab}}$ are all minimizers. There is only one length-k fragment that starts with bbb and one that starts with bba, so the first two of these final b's will both be minimizers for the windows that start with bbb and bba. For the window that starts at the third b, which is position $|T_{\mathtt{ab}}|$, note that the entire window does not contain bb at all; it consists of only alternating b's and a's as the window has length $w + k - 1$ whereas the next occurrence of bb is after $w + k + p$

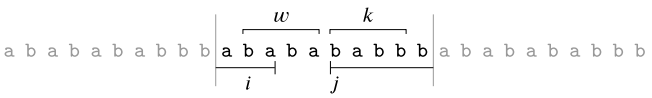


Fig. 6. $T_{\mathtt{ab}}$ for $w = 4$ and $k = 4$, showing the positions i and j when counting M_B: j is the position of the first `b` at which the corresponding length-k fragment contains `bb`; i is the last position at which j is not a candidate for its minimizer. When `b` < `a`, the minimizers in this block are all `b`'s up to position $i + 1$ and all `b`'s from position j onwards.

positions. Because the window does not contain `bb`, none of its candidates are smaller than `baba`… alternating, which first appears at the start of the window. Therefore, the third `b` is also a minimizer.

The rest of the minimizers consist of two sets. The first set corresponds to positions for which no candidate is smaller than `baba`… (alternating for k letters). These are all positions with a `b`, up to a certain position i (to be computed later), after which there will also be a smaller minimizer candidate, i.e., one that contains `bb`; inspect Fig. 6. This is the second set of minimizers: ones that start with `b` and contain `bb` at some point. These are all positions with a `b` from some position j onwards.

We start by computing j. Position j is the first position such that the length-k fragment starting at j starts with a `b` and contains `bb`. If k is odd, the fragment ends at position $2t + 2$ with `bbb` as suffix; if k is even, the fragment ends at position $2t + 1$ with `bb` as suffix. We have

$$j = \begin{cases} 2t + 1 - k + 1 = w + p + 2 & \text{if } k \text{ is even;} \\ 2t + 2 - k + 1 = w + p + 3 & \text{if } k \text{ is odd.} \end{cases}$$

Note that $j = w + p + 2 + (k \bmod 2)$. Every `b` from position j onwards is a minimizer. This includes the three `b`'s at the end of the pattern (at positions $2t$ through $2t + 2$), as well as the ones between positions j and $2t - 1$ (both inclusive). Thus we have

$$\begin{aligned} 3 + \left\lfloor \frac{2t - j}{2} \right\rfloor &= 3 + \left\lfloor \frac{w + k + p - (w + p + 2 + (k \bmod 2))}{2} \right\rfloor \\ &= 3 + \left\lfloor \frac{k - 2 - (k \bmod 2)}{2} \right\rfloor = 2 + \left\lfloor \frac{k}{2} \right\rfloor \end{aligned}$$

`b`'s from position j onwards.

Next, we compute i and count the number of `b`'s up to i. We take the last position for which the length-k fragment starting at j is not a candidate. This is $i = j - w$. The minimizer for the window starting at position $i + 1$ is the length-k fragment starting at j, since this is the only candidate that contains `bb`. However, if there is a `b` at position $i + 1$,[3] then $i + 1$ will still be a minimizer: when we take the minimizer for position i, the length-k fragment containing `bb` will not be a candidate so it will take the first length-k fragment starting with a `b`, which is at position $i + 1$. Therefore, we count all `b`'s that appear *up to* $i + 1$:

$$\begin{aligned} \left\lfloor \frac{i + 1}{2} \right\rfloor &= \left\lfloor \frac{j - w + 1}{2} \right\rfloor = \left\lfloor \frac{(w + p + 2 + (k \bmod 2)) - w + 1}{2} \right\rfloor \\ &= \left\lfloor \frac{p + 3 + (k \bmod 2)}{2} \right\rfloor = 1 + \left\lfloor \frac{1 + p + (k \bmod 2)}{2} \right\rfloor = \begin{cases} 1 + p & \text{if } k \text{ is even;} \\ 2 & \text{if } k \text{ is odd.} \end{cases} \end{aligned}$$

Adding the two numbers of `b`'s together gives (inspect Fig. 7):

$$\begin{aligned} M_B &= 2 + \left\lfloor \frac{k}{2} \right\rfloor + \begin{cases} 1 + p & \text{if } k \text{ is even;} \\ 2 & \text{if } k \text{ is odd;} \end{cases} \\ &= \begin{cases} \frac{k}{2} + 3 + p & \text{if } k \text{ is even;} \\ \left\lfloor \frac{k}{2} \right\rfloor + 4 & \text{if } k \text{ is odd.} \end{cases} \end{aligned}$$

□

2.5. Wrapping up the reduction

Proof of Lemma 1. MINIMIZING THE MINIMIZERS (DECISION) asks whether or not there exists some ordering on Σ such that a string $S \in \Sigma^n$ has at most ℓ minimizers for parameters w and k. Given w, k and an ordering on Σ, one can compute the number of minimizers

[3] Consider the case when $T_{\mathtt{ab}} = \mathtt{abababababbb}$ with $w = 5$ and $k = 4$. For this block, we have $i = 3$ and $j = 8$. Indeed $i = j - w = 3$ and at position $i + 1 = 4$ of the block we have a `b`. Position 4 will be selected as the minimizer for the window starting at position 3.

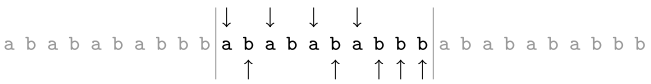


Fig. 7. $T_{\mathtt{ab}}$ for $w = 4$ and $k = 4$ (Case C), showing its minimizers for $\mathtt{a} < \mathtt{b}$ (top) and $\mathtt{b} < \mathtt{a}$ (bottom). In this situation, $M_G = 4$ and $M_B = 5$.

for those parameters in linear time [19, Theorem 3]. Therefore, one can use an alphabet ordering as a certificate to verify a YES instance of MINIMIZING THE MINIMIZERS (DECISION) simply by comparing the computed number of minimizers to ℓ. This proves that the MINIMIZING THE MINIMIZERS (DECISION) problem is in NP. To prove that MINIMIZING THE MINIMIZERS (DECISION) is NP-hard, we use a reduction from FEEDBACK ARC SET (DECISION) (see Section 2 for definition), which is a well-known NP-complete problem [18].

We are given an instance $G = (V, A)$ of FEEDBACK ARC SET and an integer ℓ', and we are asked to check if G contains a feedback arc set with at most ℓ' arcs. We will construct an instance S of MINIMIZING THE MINIMIZERS (DECISION), for given parameters $w \geq 3$ and $k \geq 1$, such that: the minimum number of minimizers in S, over all alphabet orderings, is at most some value ℓ if and only if G contains a feedback arc set of size at most ℓ'.

By Lemma 3, we have $\lambda \leq 4 \cdot |A| \cdot |T_{\mathtt{ab}}|$. Given w and k, we must determine a string $T_{\mathtt{ab}}$ such that $M_B > M_G$ and $|T_{\mathtt{ab}}| \geq \frac{1}{4}(w + k - 1)$, and also choose some q satisfying $\lambda < q \cdot (M_B - M_G)$ (see Lemma 4). Let $\Sigma = V$ and let $S = \prod_{(\mathtt{a},\mathtt{b}) \in A} T_{\mathtt{ab}}^{q+4}$, with $T_{\mathtt{ab}}$ and q to be determined depending on w and k.

Case A: $w \geq k + 2$. Let $T_{\mathtt{ab}} = \mathtt{ab}^{w-1}$, so $|T_{\mathtt{ab}}| = w$. Since, by hypothesis, the maximal value of k is $w - 2$, and since $|T_{\mathtt{ab}}| = w$, we have that $4|T_{\mathtt{ab}}| \geq 2w - 3$. Thus, the condition on the length of $T_{\mathtt{ab}}$ always holds. By Lemma 5, $M_B - M_G = w - k - 1$. We choose $q = 4 \cdot w \cdot |A| + 1$, so that $\lambda \leq 4 \cdot |A| \cdot w < q \cdot (w - k - 1)$. Thus, $\lambda < q \cdot (M_B - M_G)$.

Case B and Case C: $w < k + 2$. Let $T_{\mathtt{ab}} = (\mathtt{ab})^t\mathtt{bb}$ for $t = \left\lceil \frac{w+k}{2} \right\rceil$. We have $|T_{\mathtt{ab}}| = 2t + 2 = 2\left(\left\lceil \frac{w+k}{2} \right\rceil\right) + 2 = w + k + p + 2$, where $p = (w + k) \bmod 2$. The condition on the length of $T_{\mathtt{ab}}$ always holds because $w + k + p + 2 > w + k - 1$.

- If $w = 3$, then by Lemma 6, $M_B - M_G = \left\lfloor \frac{k}{2} \right\rfloor + 4 - \left(\left\lfloor \frac{k}{2} \right\rfloor + 3\right) = 1$.
- If $w > 3$, then by Lemma 7:
  - if k is even, then $M_B - M_G = \frac{k}{2} + 3 + p - \left(\frac{k}{2} + 2 + p\right) = 1$;
  - if k is odd, then $M_B - M_G = \left\lfloor \frac{k}{2} \right\rfloor + 4 - \left(\left\lfloor \frac{k}{2} \right\rfloor + 3\right) = 1$.

  In any case, $M_B - M_G = 1$. We choose $q = 4 \cdot |A| \cdot (w + k + 3) + 1$, so that $\lambda \leq 4 \cdot |A| \cdot (w + k + p + 2) < q$. Thus, $\lambda < q \cdot (M_B - M_G)$.

Finally, we set $\ell = q \cdot (M_B - M_G) \cdot (\ell' + 1) + q \cdot M_G \cdot |A|$. By Lemma 4, we have that $\mathcal{M}_{w,k}(S, F) \leq \ell$ if and only if $|F| \leq \ell'$; in other words, G contains a feedback arc set of size at most ℓ' if and only if S has an alphabet ordering with at most ℓ minimizers.

Hence we have shown that (G, ℓ') is a YES instance of MINIMIZING THE MINIMIZERS (DECISION) if and only if (S, ℓ) is a YES instance of FEEDBACK ARC SET (DECISION). Moreover, the length of S is $(q + 4) \cdot |A| \cdot |T_{\mathtt{ab}}|$, with $T_{\mathtt{ab}}$ being of polynomial length, so the reduction can be performed in polynomial time. The existence of a polynomial-time reduction from FEEDBACK ARC SET (DECISION) to MINIMIZING THE MINIMIZERS (DECISION) proves our claim: MINIMIZING THE MINIMIZERS (DECISION) is NP-complete if $w \geq 3$ and $k \geq 1$. □

3. MINIMIZING THE MINIMIZERS is NP-complete for $w = 2$

A weakly connected directed graph is called *Eulerian* if it contains an Eulerian circuit: a trail which starts and ends at the same graph vertex; i.e., a graph cycle which uses each arc exactly once. By Euler's famous theorem, we know that a weakly connected directed graph is Eulerian if and only if every graph vertex has equal in-degree and out-degree. In the special case when $G = (V, A)$ is a directed Eulerian graph, we call the problems FEEDBACK ARC SET and FEEDBACK ARC SET (DECISION), EULERIAN FEEDBACK ARC SET and EULERIAN FEEDBACK ARC SET (DECISION), respectively. In this section, we design a reduction from the EULERIAN FEEDBACK ARC SET (DECISION) problem to the MINIMIZING THE MINIMIZERS (DECISION) problem for the case of $w = 2$. EULERIAN FEEDBACK ARC SET (DECISION) is known to be NP-complete [24].

Given any directed Eulerian graph $G = (V, A)$, we construct a string S over alphabet $\Sigma = V$ such that the minimum number of minimizers $\mathcal{M}_{w,k}(S, F)$ for $w = 2$ can be expressed exactly as a linear function of the cardinality of the feedback arc set F. We do this by constructing a string gadget for each arc $(\mathtt{a}, \mathtt{b}) \in A$, for which we can determine the number of minimizers if $\mathtt{a} < \mathtt{b}$ and if $\mathtt{b} < \mathtt{a}$. The fact that G is Eulerian is used to allow these gadgets to *overlap*, so that every minimizer window can correspond to a single arc or vertex.

Construction of S. Let $v_0, \ldots, v_{|A|}$ denote an Eulerian circuit on the vertices of G. Every consecutive vertex pair (v_i, v_{i+1}), $i \in [0, |A|)$, corresponds to a single arc in A, and we can have no more than $|A|$ such pairs. Note however that vertices may occur multiple times

in the circuit. We construct S as follows:

$$S = \prod_{i=0}^{|A|} v_{|A|-i}^{k+1}. \tag{2}$$

Recall that the product $\prod$ of some strings denotes their concatenation, and raising a letter to some power x means repeating it x times. In other words, S is created by taking the Eulerian circuit in *reverse*, and repeating each letter v_i $k+1$ times. We call each v_i^{k+1} a *block*. By having a block of v_{i+1} followed by a block of v_i in S, we ensure that we have *one extra minimizer* when $v_{i+1} < v_i$ and several other minimizers regardless of the ordering. We reverse the arcs to ensure that this extra minimizer is added to $\mathcal{M}_{w,k}(S,F)$ when $(v_i, v_{i+1}) \in F$.

Lemma 8. $\mathcal{M}_{w,k}(S,F) = 1 + |A| \cdot k + |F|$ *if* $w = 2$.

Proof. Since $w = 2$, the windows for which minimizers are determined have length $w + k - 1 = k + 1$. By construction, for any arc (v_i, v_{i+1}), there are $k+1$ length-$(k+1)$ windows starting at some position in each block corresponding to v_{i+1}. We consider each of these windows.

The first of these windows, starting at the first v_{i+1}, is v_{i+1}^{k+1}. Since both candidates in this window all equal v_{i+1}^{k}, the leftmost fragment will be the minimizer by tie-breaking regardless of whether or not $(v_i, v_{i+1}) \in F$.

The other k windows each have the form $v_{i+1}^{m} v_i^{k+1-m}$ for $1 \le m \le k$. The first candidate of each such window starts with m v_{i+1}'s followed by v_i's, whereas the second starts with $m-1$ v_{i+1}'s followed by v_i's. Therefore, if $v_{i+1} < v_i$ (i.e., $(v_i, v_{i+1}) \in F$), the former candidate will be the window's minimizer whereas if $v_i < v_{i+1}$, the second candidate will be the minimizer. We have k of these windows, all with consecutive starting positions. The minimizer candidates starting at the third through the $(k+1)$th v_{i+1} of the block are each *both* the first candidate of some such window and the second candidate of some other such window, and will therefore be a minimizer regardless of the alphabet ordering. This amounts to $k-1$ minimizers for each arc. Out of these k windows, only the first candidate of the first window and the second candidate of the last window remain. The first candidate of the first window, which starts at the second v_{i+1} in the block, is a minimizer if and only if $v_{i+1} < v_i$. The second candidate of the last block is v_i^k, of which we already know it is a minimizer regardless of the alphabet ordering as it is the start of a new block. Thus, we have one more minimizer if $v_{i+1} < v_i$ and one more regardless of the ordering.

Finally, we have a block v_0^{k+1} at the end of S. Like any other block, the first v_0 is a minimizer by tie-breaking. The second position in this block, indicating the start of the final length-k fragment of S, is not a minimizer because it is not part of any other window.

In summary, we have:

- for every arc $(v_i, v_{i+1}) \in A$:
  - k minimizers regardless of alphabet ordering;
  - one minimizer if $(v_i, v_{i+1}) \in F$;
- one minimizer at the start of the final block, regardless of alphabet ordering.

All of this adds up to $1 + |A| \cdot k + |F|$ minimizers in total, completing the proof. □

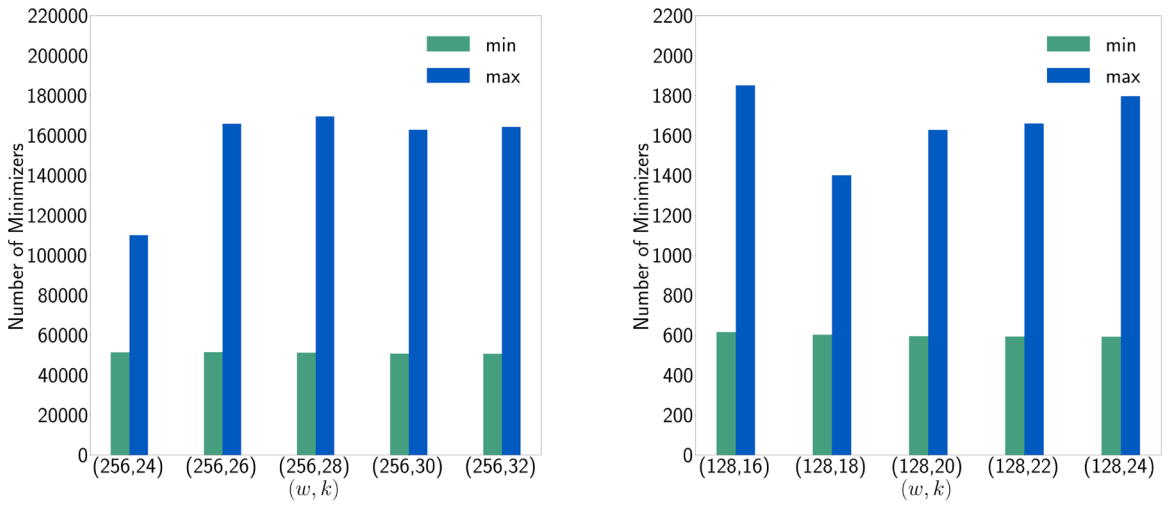
Example 3. Let $G = (V, A)$ be an Eulerian directed graph with $V = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $A = \{(\mathtt{a},\mathtt{b}), (\mathtt{b},\mathtt{c}), (\mathtt{c},\mathtt{a})\}$. Further let $\mathtt{a} < \mathtt{b} < \mathtt{c}$, $F = \{(\mathtt{c},\mathtt{a})\}$, $w = 2$, $k = 5$, and $w + k - 1 = 6$. For the circuit $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{a}$ ($\mathtt{a}, \mathtt{c}, \mathtt{b}, \mathtt{a}$ in reverse) we have $S = \mathtt{aaaaaaccccccbbbbbbaaaaaa}$ with $1 + |A| \cdot k + |F| = 17$ minimizers colored red.

Proof of Lemma 2. We show that Minimizing the Minimizers (Decision) is NP-complete if $w = 2$ by a reduction from Eulerian Feedback Arc Set (Decision). Let (G, ℓ') be an instance of Eulerian Feedback Arc Set (Decision). First, we determine an Eulerian circuit $v_0, \ldots, v_{|A|}$ of G, which is well-known to be possible in polynomial time. Using this Eulerian circuit, we construct S in accordance with Eq. (2). By Lemma 8, we know that $\mathcal{M}_{w,k}(S,F) = 1 + |A| \cdot k + |F|$. Let $\ell = 1 + |A| \cdot k + \ell'$. Because the function $\mathcal{M}_{w,k}(S,F)$ is linear in $|F|$, we have that $\mathcal{M}_{w,k}(S,F) \le \ell$ if and only if $|F| \le \ell'$. In other words, G contains a feedback arc set of size at most ℓ' if and only if there exists some alphabet ordering such that S has at most ℓ minimizers. The string S has length $(k+1) \cdot (|A| + 1)$, which is polynomial in the size of G and can therefore be constructed in polynomial time. □

Remark 1. We acknowledge the fact that the proof of Lemma 2 can be generalized to cover the case of $w \ge 3$ as well (see Section 2). In this case, we would set $S = \prod_{i=0}^{|A|} v_{|A|-i}^{w+k-1}$ and prove that $\mathcal{M}_{w,k}(S,F) = 1 + |A| \cdot k + |F| \cdot (w-1)$. However, we opted to retain the proof of Lemma 1 regardless because it is elementary; namely, it is based directly on one of Karp's original NP-complete problems [18] rather than on a special case of one. Furthermore, other than being more easily verifiable, the proof of Lemma 1 provides a lot of insight into the main problem in scope, which may inspire approximation algorithms or heuristics. Finally, the underlying technique in the proof of Lemma 1 may inspire similar hardness proofs for other sampling methods.

4. Considering the orderings on Σ^k

Most of the existing approaches for minimizing the minimizer samples consider the space of all orderings on Σ^k instead of the ones on Σ. Such an approach has the advantage of an easy and efficient construction of the sample by using a rolling hash function $h : \Sigma^k \to \mathbb{N}$, such as the popular Karp-Rabin fingerprints [25]; this results in a (pseudo-)random ordering on Σ^k that usually performs well in practice [6]. Let us denote by Minimizing the Minimizers ($\le \Sigma^k$) the version of Minimizing the Minimizers that seeks to minimize $|\mathcal{M}_{w,k}(S)|$ by choosing a best ordering on Σ^k (instead of a best ordering on Σ). It is easy to see that any algorithm



(a) Complete genome of Escherichia coli. (b) Complete genome of SARS-CoV-2.

Fig. 8. The min and max values of the size of the minimizer sample, among *some* of the possible orderings of $[1, |\Sigma|^k]$, on two real datasets using a range of (w, k) parameter values, when using the modulo sampling minimizer scheme with $t = k/2$.

solving MINIMIZING THE MINIMIZERS solves also MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) with a polynomial number of extra steps: We use an arbitrary ranking function rank from *the set* of length-k substrings of S to $[1, n-k+1]$. We construct the string S' such that $S'[i] = \mathsf{rank}(S[i \,..\, i+k-1])$, for each $i \in [1, n-k+1]$. Let Σ' be the set of all letters in S'. It should be clear that $|\Sigma'| \leq n$ because S has no more than n substrings of length k. We then solve the MINIMIZING THE MINIMIZERS problem with input $\Sigma := \Sigma'$, $S := S'$, $w := w$, and $k := 1$. It is then easy to verify that an optimal solution to MINIMIZING THE MINIMIZERS for this instance implies an optimal solution to MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) for the original instance. We thus conclude that MINIMIZING THE MINIMIZERS is at least as hard as MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$); they are clearly equivalent for $k = 1$.

Example 4. Let $S = \mathtt{aacaaacgcta}$, $w = 3$, and $k = 3$. We construct the string $S' = 235124687$ over $\Sigma' = [1, 8]$ and solve MINIMIZING THE MINIMIZERS with $w = 3$, $k = 1$, and $\Sigma = \Sigma'$. Assuming $1 < 3 < 5 < 6 < 2 < 4 < 7 < 8$, $\mathcal{M}_{3,1}(S') = \mathcal{M}_{3,3}(S) = \{2, 4, 7\}$. The minimizers positions are colored red: $S' = 23\mathbf{5}12\mathbf{4}6\mathbf{8}7$. This is one of many best orderings.

Another advantage of MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) is that a best ordering on Σ^k is at least as good as a best ordering on Σ at minimizing the resulting sample. Indeed this is because every ordering on Σ implies an ordering on Σ^k but not the reverse.

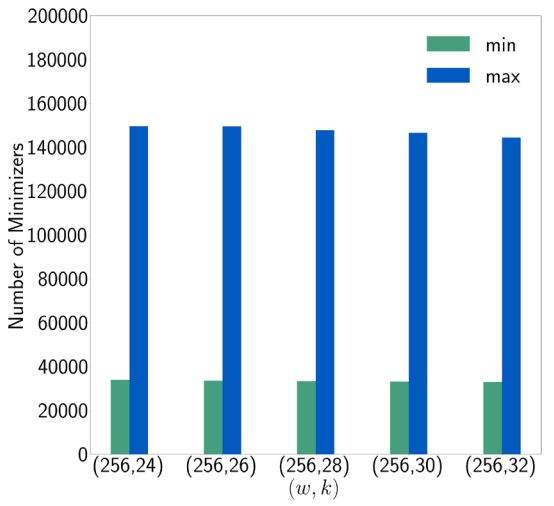
Unfortunately, MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) comes with a major disadvantage. Suppose we had an algorithm solving MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) (either exactly or with a good approximation ratio or heuristically) and applied it to a string S of length n, with parameters w and k. Now, in order to compare a query string Q to S, the first step would be to compute the minimizers of Q, but to ensure local consistency (Property 2), we would need access to the ordering output by the hypothetical algorithm. The size of the ordering is $\mathcal{O}(\min(|\Sigma|^k, n))$ and storing this defeats the purpose of creating a sketch for S [26]. This is when it might be more appropriate to use MINIMIZING THE MINIMIZERS instead. Indeed in [26], the authors make a first step toward quantifying the number of windows (size of partition) admitting the same minimizer, and provide several algorithms for realizing this computation.

Since MINIMIZING THE MINIMIZERS is NP-hard for $w \geq 2$ and $k = 1$, MINIMIZING THE MINIMIZERS ($\leq \Sigma^1$) is NP-hard for $w \geq 2$; hence the following corollary of Theorem 1.

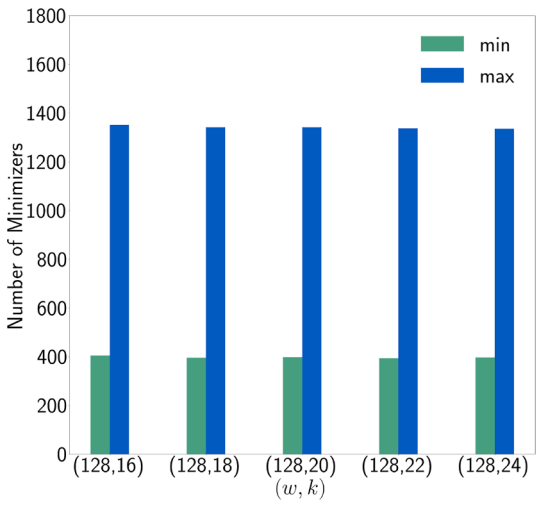
Corollary 1. *MINIMIZING THE MINIMIZERS ($\leq \Sigma^1$) is NP-hard if $w \geq 2$.*

5. Impact of reordering on minimizer schemes

In this section, we present further experimental results demonstrating the impact of alphabet reordering on other, more involved, minimizer schemes. We tested three minimizer schemes: modular-sampling [27]; miniception [14]; and rotational minimizer [28]. We used the E. coli and SARS-CoV-2 datasets from Section 1. All these schemes assume oracle access to a fixed ordering on $[1, |\Sigma|^k]$. We computed the number of minimizers for different values of (w, k) and a *subset* of the orderings on $[1, |\Sigma|^k]$. The source code for the implemented algorithms and the reported orderings for reproducing the presented results can be found at https://github.com/lorrainea/minOr.

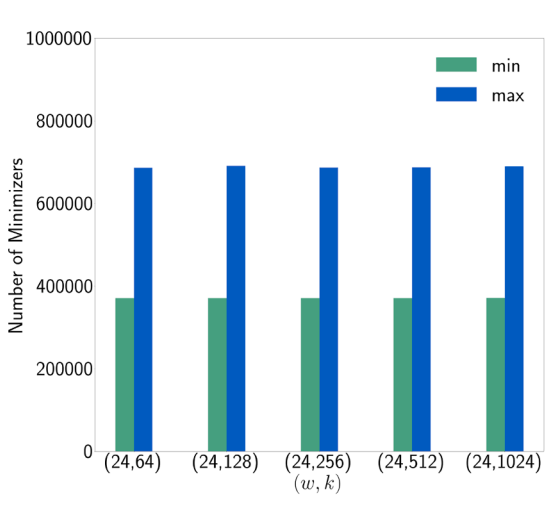


(a) Complete genome of Escherichia coli.

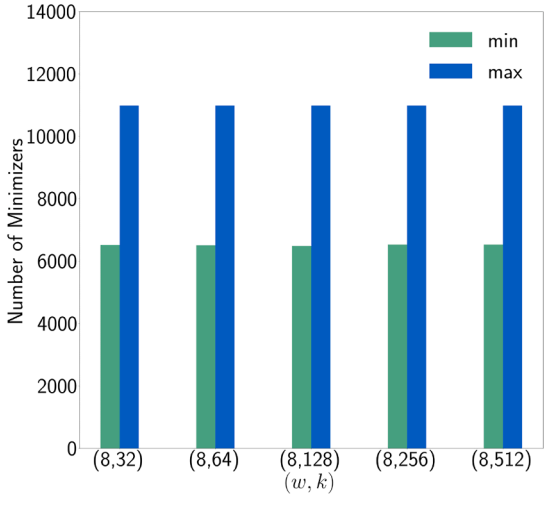


(b) Complete genome of SARS-CoV-2.

Fig. 9. The min and max values of the size of the minimizer sample, among *some* of the possible orderings of $[1, |\Sigma|^k]$, on two real datasets using a range of (w, k) parameter values, when using the miniception minimizer scheme with $t = 8$.



(a) Complete genome of Escherichia coli.



(b) Complete genome of SARS-CoV-2.

Fig. 10. The min and max values of the size of the minimizer sample, among *some* of the possible orderings of $[1, |\Sigma|^k]$, on two real datasets using a range of (w, k) parameter values, when using the rotational minimizer scheme.

Fig. 8 shows the min and max values of the size of the obtained minimizer samples when using the modular-sampling scheme [27]. This scheme finds the position x of the smallest length-t fragment, for some $t \leq k$, in a window of size $w + k - t$, and then samples the length-k fragment starting at position $p = x \bmod w$. The gap between min and max is significant as in all cases we have $2\,\text{min} < \text{max}$.

Fig. 9 shows the min and max values of the size of the obtained minimizer samples when using the miniception scheme [14]. In this scheme, there is a two-stage process. In the first one, we find the smallest length-k fragments of the window. In the second one, a second fragment of length t ($t \leq k$) is sampled in addition to that of length k. If a length-t fragment is found to occur at the start or end of a length-k fragment, then this length-k fragment is prioritized. The gap between min and max is quite significant as in all cases we have $3\,\text{min} < \text{max}$.

Fig. 10 shows the min and max values of the size of the obtained minimizer samples when using the rotational minimizer scheme [28]; the name "rotational" was coined in [27]. This scheme requires $k > w$ and considers, for each $j = [1, w]$, the sum

ψ_j of letters (with values in $[1, |\Sigma|]$) at positions $j \bmod w$ in every length-k fragment of the window. A length-k fragment is then selected when $\psi_1 + |\Sigma| \geq \max_j \psi_j$ (ties are broken by falling back to a random minimizer). The results show similar findings to before with $1.5\,\text{min} < \text{max}$ in all cases.

In summary, the above results establish, using involved minimizer schemes, that the impact of alphabet reordering is significant (e.g., $3\,\text{min} < \text{max}$ holds in many cases).

6. Final remarks

In this paper, we considered the MINIMIZING THE MINIMIZERS problem: Given a string S over an alphabet Σ and two integers w and k, can we efficiently compute a total order on Σ that minimizes the minimizers of S? We concluded that this is unlikely by showing that the problem is NP-hard for any non-trivial values of w and k.

We leave the following three questions unanswered:

1. Is there a more efficient exponential-time algorithm for MINIMIZING THE MINIMIZERS, e.g. running in $\mathcal{O}(n \cdot c^{|\Sigma|})$ time, for a constant c, instead of the naïve $\mathcal{O}(n \cdot |\Sigma|!)$-time algorithm that enumerates all possible orderings?
2. Can we design an approximation scheme for MINIMIZING THE MINIMIZERS?
3. Is the problem of choosing a best ordering on Σ^k (instead of a best ordering on Σ) any easier? Namely, is MINIMIZING THE MINIMIZERS ($\leq \Sigma^k$) NP-hard for $k > 1$?

CRediT authorship contribution statement

Hilde Verbeek: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization; **Lorraine A.K. Ayad:** Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization; **Grigorios Loukides:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation; **Solon P. Pissis:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Solon P. Pissis reports financial support was provided by European Commission. Solon P. Pissis reports a relationship with European Commission that includes: funding grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Schleimer, D.S. Wilkerson, A. Aiken, Winnowing: local algorithms for document fingerprinting, in: A.Y. Halevy, Z.G. Ives, A. Doan (Eds.), Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9–12, 2003, ACM, New York, NY, USA, 2003, pp. 76–85. https://doi.org/10.1145/872757.872770

[2] M. Roberts, W.B. Hayes, B.R. Hunt, S.M. Mount, J.A. Yorke, Reducing storage requirements for biological sequence comparison, Bioinform. 20 (18) (2004) 3363–3369. https://doi.org/10.1093/bioinformatics/bth408

[3] H. Li, Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences, Bioinform. 32 (14) (2016) 2103–2110. https://doi.org/10.1093/BIOINFORMATICS/BTW152

[4] H. Li, Minimap2: pairwise alignment for nucleotide sequences, Bioinform. 34 (18) (2018) 3094–3100. https://doi.org/10.1093/BIOINFORMATICS/BTY191

[5] D.E. Wood, S.L. Salzberg, Kraken: ultrafast metagenomic sequence classification using exact alignments, Genome Biol. 15 (3) (2014) R46. https://doi.org/10.1186/gb-2014-15-3-r46

[6] H. Zheng, G. Marçais, C. Kingsford, Creating and using minimizer sketches in computational genomics, J. Comput. Biol. 30 (12) (2023) 1251–1276. https://doi.org/10.1089/CMB.2023.0094

[7] S. Grabowski, M. Raniszewski, Sampled suffix array with minimizers, Softw. Pract. Exp. 47 (11) (2017) 1755–1771. https://doi.org/10.1002/SPE.2481

[8] Y. Shibuya, D. Belazzougui, G. Kucherov, Space-efficient representation of genomic k-mer count tables, Algorithms Mol. Biol. 17 (1) (2022) 5. https://doi.org/10.1186/S13015-022-00212-0

[9] G. Loukides, S.P. Pissis, M. Sweering, Bidirectional string anchors for improved text indexing and top-k similarity search, IEEE Trans. Knowl. Data Eng. 35 (11) (2023) 11093–11111. https://doi.org/10.1109/TKDE.2022.3231780

[10] L.A.K. Ayad, G. Loukides, S.P. Pissis, Text indexing for long patterns: anchors are all you need, Proc. VLDB Endow. 16 (9) (2023) 2117–2131. https://www.vldb.org/pvldb/vol16/p2117-loukides.pdf. https://doi.org/10.14778/3598581.3598586

[11] R. Chikhi, A. Limasset, S. Jackman, J.T. Simpson, P. Medvedev, On the representation of De Bruijn graphs, J. Comput. Biol. 22 (5) (2015) 336–352. https://doi.org/10.1089/CMB.2014.0160

[12] S. Deorowicz, M. Kokot, S. Grabowski, A. Debudaj-Grabysz, KMC 2: fast and resource-frugal *k*-mer counting, Bioinform. 31 (10) (2015) 1569–1576. https://doi.org/10.1093/BIOINFORMATICS/BTV022

[13] Y. Orenstein, D. Pellow, G. Marçais, R. Shamir, C. Kingsford, Compact universal k-mer hitting sets, in: M.C. Frith, C.N.S. Pedersen (Eds.), Algorithms in Bioinformatics - 16th International Workshop, WABI 2016, Aarhus, Denmark, August 22–24, 2016. Proceedings, 9838 of *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, 2016, pp. 257–268. https://doi.org/10.1007/978-3-319-43681-4_21

[14] H. Zheng, C. Kingsford, G. Marçais, Improved design and analysis of practical minimizers, Bioinform. 36 (Supplement-1) (2020) i119–i127. https://doi.org/10.1093/BIOINFORMATICS/BTAA472

[15] C. Jain, A. Rhie, H. Zhang, C. Chu, B. Walenz, S. Koren, A.M. Phillippy, Weighted minimizer sampling improves long read mapping, Bioinform. 36 (Supplement-1) (2020) i111–i118. https://doi.org/10.1093/BIOINFORMATICS/BTAA435

[16] H. Zheng, C. Kingsford, G. Marçais, Sequence-specific minimizers via polar sets, Bioinform. 37 (Supplement) (2021) 187–195. https://doi.org/10.1093/BIOINFORMATICS/BTAB313

[17] M. Hoang, H. Zheng, C. Kingsford, Differentiable learning of sequence-specific minimizer schemes with deepminimizer, J. Comput. Biol. 29 (12) (2022) 1288–1304. https://doi.org/10.1089/CMB.2022.0275
[18] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, Plenum Press, New York, NY, USA, 1972, pp. 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9
[19] G. Loukides, S.P. Pissis, Bidirectional string anchors: a new string sampling mechanism, in: P. Mutzel, R. Pagh, G. Herman (Eds.), 29th Annual European Symposium on Algorithms, ESA 2021, September 6–8, 2021, Lisbon, Portugal (Virtual Conference), 204 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 64:1–64:21. https://doi.org/10.4230/LIPICS.ESA.2021.64
[20] J.W. Bentley, D. Gibney, S.V. Thankachan, On the complexity of BWT-Runs minimization via alphabet reordering, in: F. Grandoni, G. Herman, P. Sanders (Eds.), 28th Annual European Symposium on Algorithms, ESA 2020, September 7–9, 2020, Pisa, Italy (Virtual Conference), 173 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2020, pp. 15:1–15:13. https://doi.org/10.4230/LIPICS.ESA.2020.15
[21] D. Gibney, S.V. Thankachan, Finding an optimal alphabet ordering for lyndon factorization is hard, in: M. Bläser, B. Monmege (Eds.), 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16–19, 2021, SaarbrÜCken, Germany (Virtual Conference), 187 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 35:1–35:15. https://doi.org/10.4230/LIPICS.STACS.2021.35
[22] H. Verbeek, L.A.K. Ayad, G. Loukides, S.P. Pissis, Minimizing the minimizers via alphabet reordering, in: S. Inenaga, S.J. Puglisi (Eds.), 35th Annual Symposium on Combinatorial Pattern Matching, CPM 2024, June 25–27, 2024, Fukuoka, Japan, 296 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 28:1–28:13. https://doi.org/10.4230/LIPICS.CPM.2024.28
[23] D.H. Younger, Minimum feedback arc sets for a directed graph, IEEE Trans. Circuit Theory 10 (2) (1963) 238–245. https://doi.org/10.1109/TCT.1963.1082116
[24] K. Perrot, V.T. Pham, Feedback arc set problem and NP-hardness of minimum recurrent configuration problem of chip-firing game on directed graphs, Ann. Comb. 19 (2015) 373–396. https://doi.org/10.1007/s00026-015-0266-9
[25] R.M. Karp, M.O. Rabin, Efficient randomized pattern-matching algorithms, IBM J. Res. Dev. 31 (2) (1987) 249–260. https://doi.org/10.1147/RD.312.0249
[26] F. Ingels, C. Marchet, M. Salson, On the number of *k*-mers admitting a given lexicographical minimizer, CoRR, (2024). https://doi.org/10.48550/ARXIV.2412.17492
[27] R. Groot Koerkamp, G.E. Pibiri, The mod-minimizer: a simple and efficient sampling algorithm for long k-mers, in: S.P. Pissis, W.-K. Sung (Eds.), 24th International Workshop on Algorithms in Bioinformatics (WABI 2024), 312 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 11:1–11:23. https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.WABI.2024.11. https://doi.org/10.4230/LIPIcs.WABI.2024.11
[28] G. Marçais, D. DeBlasio, C. Kingsford, Asymptotically optimal minimizers schemes, Bioinformatics 34 (13) (2018) i13–i22. https://doi.org/10.1093/bioinformatics/bty258