Theoretical Computer Science 978 (2023) 114182

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Maximal degenerate palindromes with gaps and mismatches [☆]

Mai Alzamel [a], Christopher Hampson [b], Costas S. Iliopoulos [b], Zara Lim [b,*], Solon Pissis [c,e], Dimitrios Vlachakis [d], Steven Watts [b]

[a] Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, 145111, Saudi Arabia
[b] Department of Informatics, King's College London, 30 Aldwych, London, WC2B 4BG, UK
[c] Department of Computer Science, Vrije Universiteit, Amsterdam, Netherlands
[d] Department of Biotechnology, Agricultural University of Athens, Athens, Greece
[e] Networks and Optimization, Centrum Wiskunde & Informatica, Amsterdam, Netherlands

ARTICLE INFO

Article history:
Received 15 May 2023
Received in revised form 31 August 2023
Accepted 8 September 2023
Available online 15 September 2023

Keywords:
Degenerate
Palindrome
String factorisation
Inverted repeat
Mismatch

ABSTRACT

A degenerate symbol over an alphabet Σ is a non-empty subset of Σ, and a sequence of such symbols is a degenerate string. We investigate the exact computation of maximal degenerate palindromes with gaps and mismatches. We present an algorithm which, given a degenerate string of length n and natural number parameters g and m, efficiently detects exact maximal palindromes with a gap size $\leq g$, and $\leq m$ permitted mismatches. We show that it can be done in $O(k|\Sigma|(k + \log|\Sigma|) + (k + g + m)n)$ time and $O((g + m)n)$ space, where k represents an upper bound on the number of degenerate symbols contained in the string. Furthermore, we also show that the problem of factorisation a string into maximal degenerate palindromes with gaps and mismatches can also be done in $O(k|\Sigma|(k + \log|\Sigma|) + (k + g + m)n)$ time and $O((g + m)n)$ space. An inverted repeat is a specific type of palindrome which refers to a nucleotide sequence followed by its reverse complement. Our results can also be used to find maximal inverted repeated sequences with gaps and mismatches, where changing the structure of palindromes to inverted repeats does not affect the overall running time. Finally we demonstrate our algorithm on several strains of SARS-CoV-2, and quantify the number of inverted repeats found with $\leq 0, 1, 2$ mismatches and $\leq 0, 10, 100$ gap size.

1. Introduction

The current pandemic of COVID-19 has led to over 765 million confirmed cases and over 6.9 million fatalities globally. COVID-19 is a respiratory disease caused by a novel virus strain called SARS-CoV-2. SARS-CoV-2 is an enveloped, positive-sense, single-stranded RNA betacoronavirus of the Coronaviridae family. SARS-CoV-2 has a genome size of approximately 30 Kb, which is one of the longest RNA virus genomes known, setting several difficulties for bioinformatics research.

The occurrence of palindromes naturally arise in coronaviruses, appearing either as a standard palindrome or as an Inverted Repeat. An Inverted Repeat (IR) refers to a nucleotide sequence that is followed by its reverse complement, which naturally occurs across all organisms, and contributes towards important biological functions. If the IR contains a nucleotide sequence occurring between the first half and reverse complement of the IR, this sequence is defined as a *gap* or *spacer*.

[☆] This article belongs to Section C: Theory of natural computing, Edited by Lila Kari.
[*] Corresponding author.
E-mail address: zara.lim@kcl.ac.uk (Z. Lim).

The gap can have any length, including zero, in which case, the complete IR sequence is defined as a *palindromic sequence*. In this instance, the sequence of nucleotides read from 5' to 3' in the forward direction will be identical as the sequence from 5' to 3' on the complementary strand.

IRs can form hairpin stem-loops or cruciform structures, which are specific instances of secondary structures that occur within double-stranded nucleic acid. Such structures are associated with genomic instability, inhibiting DNA replication, and genomic activities that result in mutations [1–3]. In such structures, the nucleotides from one half of the IR base-pair with the complementary nucleotide in the corresponding position of the other half, forming the "stem", and the gap between each half of the inverted repeat corresponds to the "loop".

Other studies also suggest that inverted repeats play a role in viral packaging, replication and defence mechanisms [4–9]. Palindromes of length-4 are underrepresented across coronaviruses, where SARS additionally lacks length-6 palindromes [10]. This behaviour can be attributed to a common occurrence in bacterial and phage genomes to prevent restriction enzymes from splitting DNA molecules in palindromic regions [7,11–13].

1.1. *Algorithms for computing palindromes and gapped palindromes*

Palindrome detection is a well-studied problem in computer science, language theory and algorithm design in particular, which has yielded a range of variants from different practical scenarios. In 1975, Manacher discovered an on-line algorithm that finds all palindromes occurring at the start of a string of length n in $\mathcal{O}(n)$ time [14]. Apostolico et al. later observed that Manacher's initial palindrome algorithm can in fact locate all maximal palindromic substrings (palindromes) in the string in $\mathcal{O}(n)$ time [15]. Gusfield presented another $\mathcal{O}(n)$-time algorithm which finds all maximal palindromes, and discussed the relation between biological sequences and gapped (separated) palindromes (i.e. strings of the form $xv\overline{x}^R$) [16]; a string x is followed by a string v (that is the gap) followed by the inverted repeat of the complement of x, $\overline{x}^R$. Gupta et al. presented an $\mathcal{O}(n)$-time algorithm to compute specific classes—based on length constraints of such palindromes [17]. Algorithms for finding gapped palindromes were also considered in [18,19]. More recent variations are the palindromic factorisation problem, which determines the minimal number of palindromes a string can be decomposed into [20–26], as well as palindrome detection within weighted [27] strings.

1.2. *Degenerate strings*

A degenerate symbol over an alphabet Σ is a non-empty subset of Σ. A degenerate string over an alphabet Σ is a sequence degenerate symbols. Degenerate strings have been significantly researched (see [28–32]). More recent variants investigate elastic degenerate strings, whereby any subset does not contain, in general, only letters; a set may also contain strings of arbitrary length [33–36]. Such sequences have useful applications within biology and have been used extensively to model DNA sequences with uncertainties [37] as an alternative to graphs [38].

1.3. *Our contributions*

In this paper, we describe algorithms which, given a degenerate string of length n containing k degenerate symbols, detect maximal degenerate palindromes containing a gap with maximum size g and up to m errors and show that they can all be computed in $\mathcal{O}(k|\Sigma|(k+\log|\Sigma|)+(k+g+m)n)$ time and $\mathcal{O}((g+m)n)$ space. We also show how our algorithms can be adapted to locate inverted repeats in degenerate strings with a gap of up to g symbols and m mismatches in the same time and space complexity. Furthermore, we also show that we can decompose a degenerate string into adjacent factors, where all of the factors are maximal palindromes and a minimal number of palindromes is used, in $\mathcal{O}(k|\Sigma|(k+\log|\Sigma|)+(k+g+m)n)$ time and $\mathcal{O}((g+m)n)$ space. Finally, we conclude with a discussion about how our results may be used to improve methods for secondary structure predictions.

2. Preliminaries

Let $S=S[1]S[2]\mathinner{.\,.}S[n]$ be a string of length $|S|=n$ over an alphabet Σ. We consider the case of an integer alphabet; in this case each letter can be replaced by its rank so that the resulting string consists of integers in the range $\{1,\ldots,n\}$. For two positions i and j in S, where $1\le i\le j\le n$, we denote the factor $S[i]S[i+1]\ldots S[j]$ of S by $S[i\mathinner{.\,.}j]$. We denote the reverse string of S by S^R, i.e. $S^R=S[n]S[n-1]\ldots S[1]$. The empty string (denoted by ε) is the unique string over Σ of length 0. A string S is said to be a *palindrome* if and only if $S=S^R$. If $S[i\mathinner{.\,.}j]$ is a palindrome, the number $\frac{i+j}{2}$ is called the centre of $S[i\mathinner{.\,.}j]$. Let $S[i\mathinner{.\,.}j]$, where $1\le i\le j\le n$, be a palindromic factor in S. It is said to be a maximal palindrome, if there is no longer palindrome in S with centre $\frac{i+j}{2}$. Note that a maximal palindrome can be a factor of another palindrome. For definitions of maximal palindromic decomposition, see [22].

Note that any single letter is a palindrome and, hence, every string can always be decomposed into palindromes. However, not every string can be decomposed into maximal palindromes; e.g. consider $S=abaca$, which has maximal palindromes aba and aca [22].

Let f be an involution on the alphabet Σ, i.e., a function such that $f^2=\text{id}$, where id is the identity function which returns its argument value unchanged. We extend f into a morphism on strings over Σ. We say that a string x is a generalized palindrome if $x=f(x^R)$. Two known notions fit this definition:

$$\begin{pmatrix}\mathtt{A}\\\mathtt{C}\end{pmatrix}\mathtt{AG}\begin{pmatrix}\mathtt{C}\\\mathtt{G}\end{pmatrix}\mathtt{A}\begin{pmatrix}\mathtt{A}\\\mathtt{C}\\\mathtt{G}\end{pmatrix}$$

Fig. 1. Degenerate string of length $n = 6$ over $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ with 3 degenerate symbols.

- If $f = \mathrm{id}$, then a generalized palindrome is a standard palindrome.
- If $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and $f(\mathtt{A}) = \mathtt{T}$, $f(\mathtt{C}) = \mathtt{G}$, $f(\mathtt{G}) = \mathtt{C}$, $f(\mathtt{T}) = \mathtt{A}$, then a generalized palindrome corresponds to an inverted repeat or so-called complemented palindrome [16].

Example 1. The string AGTACTTCATGA is a standard palindrome and the string TAGTCGACTA is an inverted repeat.

We also consider (generalized) palindromes with errors. The *Hamming distance* between two equal-length strings X and Y is defined as the number of corresponding locations in X and Y with different characters, denoted $\delta_H(X, Y) = |\{i : X[i] \neq Y[i], i = 0, 1, \dots, |X| - 1\}|$. If $|X| \neq |Y|$, we set $\delta_H(X, Y) = \infty$. If two strings X and Y are at Hamming distance m or less, we call this an *m-match*, written as $X \approx_m Y$.

We say that x is a *generalized δ-palindrome* under the Hamming distance if the minimum Hamming distance from x to any generalized palindrome is *at most* δ. A generalized palindrome $S[i \dots j]$ is called maximal if there is no longer generalized palindrome with the same centre. Similarly, a generalized δ-palindrome $S[i \,..\, j]$ under the Hamming distance is called maximal if there is no longer generalized δ-palindrome under the same distance measure with the same centre.

Example 2. All maximal 0-palindromes/1-palindromes in GTATCG (for f = id) under the Hamming distance are as follows:

centre	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6
0	G	ε	T	ε	TAT	ε	T	ε	C	ε	G
1 under Hamming	G	GT	GTA	TA	GTATC	AT	ATC	TC	TCG	CG	G

2.1. Degenerate strings

We use IUPAC-encoded strings as an example for degenerate strings (see Table 1). A *degenerate symbol* $\tilde{x}$ over an alphabet Σ is a non-empty subset of Σ, i.e. $\tilde{x} \subseteq \Sigma$ and $\tilde{x} \neq \emptyset$. $|\tilde{x}|$ denotes the size of the set and we have $1 \leq |\tilde{x}| \leq |\Sigma|$. A finite sequence $\tilde{X} = \tilde{x}_0\tilde{x}_1 \dots \tilde{x}_{n-1}$ is said to be a *degenerate string* (also known as an *indeterminate string*) if $\tilde{x}_i$ is a degenerate symbol for each $0 \leq i \leq n - 1$. A degenerate string is built over the potential $2^{|\Sigma|} - 1$ non-empty subsets of characters belonging to Σ. The *length* of a degenerate string $\tilde{X}$ is the number of degenerate symbols n.

For example, $\tilde{X} = [\mathtt{A}, \mathtt{C}][\mathtt{A}][\mathtt{G}][\mathtt{C}, \mathtt{G}][\mathtt{A}][\mathtt{A}, \mathtt{C}, \mathtt{G}]$ is a degenerate string of length 6 over the alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}\}$ (or $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ with no occurrences of T). If $|\tilde{x}_i| = 1$, that is $\tilde{x}$ represents a single character of Σ, we say that $\tilde{x}_i$ is a *solid symbol* and i is a *solid position*. Otherwise $\tilde{x}_i$ and i are said to be a *non-solid symbol* and *non-solid position* respectively. For convenience we often write $\tilde{x}_i = \sigma$ ($\sigma \in \Sigma$), instead of $\tilde{x}_i = [\sigma]$, in the case of solid symbols. Consequently, the previous example $\tilde{X}$ may be written as $\tilde{X} = [\mathtt{A}, \mathtt{C}]\mathtt{AG}[\mathtt{C}, \mathtt{G}]\mathtt{A}[\mathtt{A}, \mathtt{C}, \mathtt{G}]$. A degenerate string containing only solid symbols is a *solid string* and behaves the same as a classical string of characters, and for such strings we may omit the $\sim$ notation. In addition, a solid symbol $[\sigma]$ and its corresponding character $\sigma \in \Sigma$ may be treated as interchangeable for our purposes.

The concatenation of degenerate strings $\tilde{X}$ and $\tilde{Y}$ is $\tilde{X}\tilde{Y}$. A degenerate string $\tilde{V}$ is a substring of a degenerate string $\tilde{X}$ if $\tilde{X} = \tilde{U}\tilde{V}\tilde{W}$ for some degenerate strings $\tilde{U}$ and $\tilde{W}$. By $\tilde{X}[i \,..\, j]$ we represent a substring $\tilde{x}_i\tilde{x}_{i+1} \dots \tilde{x}_j$ of $\tilde{X}$.

For degenerate strings, the notion of character equality is extended to symbol equality between two degenerate symbols. A degenerate symbol $\tilde{x}$ over an alphabet Σ is a nonempty subset of Σ, i.e. $\tilde{x} \subseteq \Sigma$ and $\tilde{x} \neq \emptyset$. $|\tilde{x}|$ denotes the size of the set and we have $1 \leq |\tilde{x}| \leq |\Sigma|$. Two degenerate symbols $\tilde{x}$ and $\tilde{y}$ are said to *match* (denoted $\tilde{x} \approx \tilde{y}$) if $\tilde{x} \cap \tilde{y} \neq \emptyset$. Extending this notion to degenerate strings, we say that two degenerate strings $\tilde{X}$ and $\tilde{Y}$ match (denoted $\tilde{X} \approx \tilde{Y}$) if $|\tilde{X}| = |\tilde{Y}|$ and all corresponding symbols in $\tilde{X}$ and $\tilde{Y}$ match. Note that the relation $\approx$ is not transitive. A degenerate string $\tilde{X}$ is said to *occur* at position i in another degenerate string $\tilde{Y}$ if $\tilde{X} \approx \tilde{Y}[i \,..\, i + |\tilde{X}| - 1]$.

We demonstrate an example of degenerate string notation in Fig. 1, where a vector of characters is used to indicate the possible values of a non-solid symbol.

2.2. Palindromes

A *palindromic factor* of a degenerate string $\tilde{X}$ is some solid string $P \approx \tilde{X}[i \,..\, j]$ such that P is equal to its reversal ($P = P^R$). Equivalently we can define an even length palindromic factor P as a factor that can be expressed in the form WW^R for some string W, and an odd length palindromic factor P as a factor that can be expressed in the form WcW^R for some string W and a single character c. The *centre* of a palindromic factor $P \approx \tilde{X}[i \,..\, j]$ is defined as $\frac{i+j}{2}$, and its *radius* is defined as $\frac{|P|}{2}$.

$$\binom{\mathtt{A}}{\mathtt{G}}\mathtt{ACG}\binom{\mathtt{A}}{\mathtt{C}}\binom{\mathtt{C}}{\mathtt{G}}\mathtt{GAC}\underbrace{\binom{\mathtt{A}}{\mathtt{G}}\binom{\mathtt{A}}{\mathtt{G}}\overset{\blacktriangledown}{\mathtt{C}}\binom{\mathtt{C}}{\mathtt{G}}\mathtt{G}}_{\mathtt{GGCGG}}\mathtt{G}\binom{\mathtt{A}}{\mathtt{C}}\mathtt{ACGCA}\binom{\mathtt{A}}{\mathtt{C}}\mathtt{CA}$$

Fig. 2. Maximal palindrome for centre 11.

Table 1
(a) IUPAC encodings as subsets of {A, C, G, T}, and (b) corresponding matching table for complementary bases.

IUPAC code	Equivalent subset
R	{A, G}
Y	{C, T}
S	{C, G}
W	{A, T}
K	{G, T}
M	{A, C}
B	{C, G, T}
D	{A, G, T}
H	{A, C, T}
V	{A, C, G}
N	{A, C, G, T}

(a)

	A	C	G	T	R	Y	S	W	K	M	B	D	H	V	N
A	1				1			1		1		1	1	1	1
C		1				1	1			1	1		1	1	1
G			1		1		1		1		1	1		1	1
T				1		1		1	1		1	1	1		1
R	1		1		1		1	1	1	1	1	1	1	1	1
Y		1		1		1	1	1	1	1	1	1	1	1	1
S		1	1		1	1	1		1	1	1	1	1	1	1
W	1			1	1	1		1	1	1	1	1	1	1	1
K			1	1	1	1	1	1	1		1	1	1	1	1
M	1	1			1	1	1	1		1	1	1	1	1	1
B		1	1	1	1	1	1	1	1	1	1	1	1	1	1
D	1		1	1	1	1	1	1	1	1	1	1	1	1	1
H	1	1		1	1	1	1	1	1	1	1	1	1	1	1
V	1	1	1		1	1	1	1	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(b)

A *maximal palindromic factor* of a degenerate string $\tilde{X}$ is the longest palindromic factor at a given centre, which can be extended no further. In other words $P \approx \tilde{X}[i\,..\,j]$ is maximal for its centre $\frac{i+j}{2}$ if $\tilde{X}[i-1] \not\approx \tilde{X}[j+1]$ or if either $i-1$ or $j+1$ are invalid indexes of $\tilde{X}$.

For example in Fig. 2 we show a palindromic factor at centre 11 corresponding to $\tilde{X}[9\,..\,13]$. In fact this is a maximal palindromic factor, as it can not be extended further within $\tilde{X}$ while still containing a palindrome, i.e. $\tilde{X}[8] = \mathtt{C} \not\approx \mathtt{G} = \tilde{X}[14]$.

We denote by $MP(\tilde{X})$ the set of all maximal palindromes of $\tilde{X}$, i.e. the set containing all pairs (i, j) such that $\tilde{X}[i\,..\,j]$ is a maximal palindrome in $\tilde{X}$. By considering the number of valid centres, it is clear that the number of unique maximal palindromic factors must be no more than $2n-1$, i.e. $|MP(\tilde{X})| \leq 2n-1$.

2.3. Longest common extensions

The *longest common extension* (LCE) between two suffixes of a string X starting at positions i and j, is defined as the length of the longest prefix common to both suffixes. We state this formally in Definition 1.

Definition 1. For a given string X, we define the *longest common extension* between position i and j as the function:

$$LCE(X, i, j) = \max(\{l : X[i\,..\,i+l-1] = X[j\,..\,j+l-1]\} \cup \{0\}).$$

We may generalise Definition 1 to that of an LCE with m mismatches. In this case, the LCE is similarly defined, but considers two prefixes to constitute a common prefix if they match within m or less errors in terms of their Hamming distance.

Definition 2. For a given string X, we define the *longest common extension with m mismatches* between position i and j as the function:

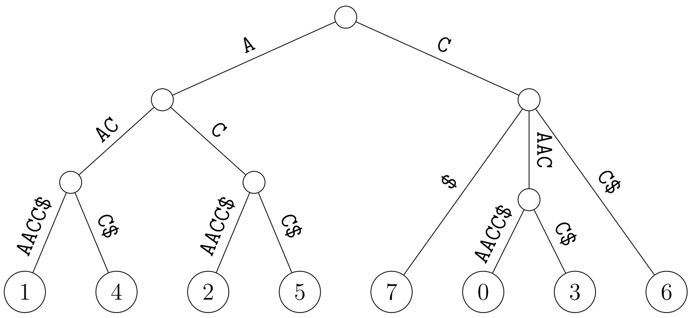
$$LCE_m(X, i, j) = \max(\{l : X[i\,..\,i+l-1] \approx_m X[j\,..\,j+l-1]\} \cup \{0\}).$$

A further variant of LCE makes use of *matching tables*. We define a *matching table* $\mathcal{M}$ over a given alphabet Σ, as any commutative binary function mapping pairs of characters of Σ to the set $\{\mathtt{true}, \mathtt{false}\}$, i.e.

$$\mathcal{M} : \Sigma \times \Sigma \to \{\mathtt{true}, \mathtt{false}\}.$$

For a given alphabet Σ, if a matching table is not explicitly defined, we may assume by default that $\mathcal{M}(s_1, s_2)$ maps to the logical result of $s_1 = s_2$ for all $s_1, s_2 \in \Sigma$. In the case of inverted repeats, a match is assigned true for complementary bases (see Table 1).

We may define the LCE with respect to a matching table $\mathcal{M}$, as in Definition 3. In this case, the LCE considers the prefixes starting at i and j to match if their corresponding characters match with respect to the matching table, i.e. two characters s_1 and s_2 match if $\mathcal{M}(s_1, s_2) = \mathtt{true}$.



	0	1	2	3	4	5	6	7	8
x	C	A	A	C	A	A	C	C	$

Fig. 3. Suffix tree for the string $X = \texttt{caacaacc\$}$.

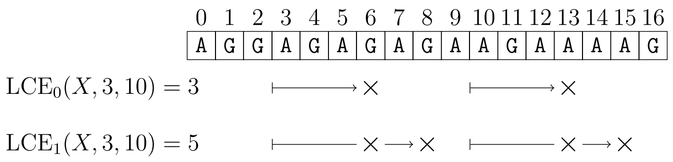


Fig. 4. Longest common extensions of a string for $m = 0$ and $m = 1$.

Definition 3. For a given string X over the alphabet Σ and a matching table $\mathcal{M}$ over Σ, we define the *longest common extension with respect to* $\mathcal{M}$ between position i and j as the function:

$$LCE(X, i, j, \mathcal{M}) = \max(\{l : \mathcal{M}(X[i + l' - 1], X[j + l' - 1]) = \texttt{true} \\ \forall l' \in [1, l]\} \cup \{0\}).$$

2.4. Kangaroo method

The algorithm described in this paper makes substantial use of the *kangaroo method*, a well established method used to perform multiple LCE_m queries on a given string X [39,40]. This is done by initially preprocessing the string X to build a *suffix tree* data structure in $\mathcal{O}(n)$ time and space [41].

The suffix tree provides a compact representation of the set of suffixes of the string X, in which each leaf node of the tree uniquely corresponds to one of the n suffixes of X, which may be reconstructed by following the unique path from the root to the leaf and concatenating the edge labels as they are encountered. To ensure every suffix corresponds uniquely to a leaf node, the string X may be appended with a unique \$ character. Full technical details of the suffix tree data structure can be found in [41]. (See Fig. 3.)

The suffix tree of X allows us to perform LCE queries in $\mathcal{O}(1)$ time. For a query $LCE(X, i, j)$, we first identify two distinct leaf nodes corresponding to the suffixes i and j; then, the lowest common ancestor node in the tree for these two leaf nodes has a string depth equal to $LCE(X, i, j)$. The calculation of the lowest common ancestor and its depth may be performed in $\mathcal{O}(1)$ time after $\mathcal{O}(n)$ preprocessing time [42], hence the LCE query can be computed in $\mathcal{O}(1)$ time.

The kangaroo method extends this methodology, allowing the calculation of LCE_m queries. Precisely, we have Lemma 1.

Lemma 1. *Given the suffix tree of X, $LCE_m(X, i, j)$ can be computed in $\mathcal{O}(m)$ time.*

Proof. $LCE_m(X, i, j)$ can be defined recursively as follows:

We denote $l_r = LCE_r(X, i, j)$ for any $r \geq 0$. Then, we have $l_0 = LCE(X, i, j)$. Also, we have $l_r = l_{r-1} + 1 + LCE(X, i + l_{r-1} + 1, j + l_{r-1} + 1)$. (See Fig. 4.)

By the above recursive formula, $LCE_m(X, i, j)$ can be computed by performing LCE queries m times. Since each LCE query requires $\mathcal{O}(1)$ time, the lemma follows. $\square$

3. Algorithms

3.1. Preprocessing

We first preprocess the string $\tilde{X}$, to obtain a string we call the *solid equivalent* of $\tilde{X}$, denoted $X_\$$. To obtain $X_\$$ we transform $\tilde{X}$ into a solid string by replacing each of its k non-solid symbols with a unique character $\$_d$, where $0 \le d < k$. These new characters $\$_0, ..., \$_{k-1}$ are defined to not match each other and not match any characters of the original alphabet Σ of $\tilde{X}$. The location in $X_\$$ of a given $\$_d$ is denoted by $\texttt{loc}(\$_d)$.

$$X_\$[i] = \begin{cases} \tilde{X}[i] & \text{if } \tilde{X}[i] \text{ is solid} \\ \$_d & \text{if } \tilde{X}[i] \text{ is non-solid and } d \text{ non-solid symbols occur left of } \tilde{X}[i] \end{cases}$$

We define the *location* $\texttt{loc}(\$_d)$ of the non-solid symbol $\$_d$ in the string $X_\$$ to be the (unique) integer $t \le n$, such that

$$\texttt{loc}(\$_d) = t \qquad \iff \qquad X_\$[t] = \$_d.$$

Let $\Sigma^{\$\#} = \Sigma \cup \{\$_0, \dots, \$_{k-1}\} \cup \{\#_1, \#_2\}$ denote the alphabet Σ extended by $\$_0, \dots, \$_{k-1}$ in addition to two new distinct characters $\#_1$ and $\#_2$, and extend the lexicographical ordering on Σ such that $\#_1 < \#_2 < \$_1 < \cdots < \$_{k-1} < \sigma_0 < \cdots < \sigma_{|\Sigma|-1}$, where σ_i refers to the ith character in the lexicographical ordering of the alphabet Σ (0-indexed). Next we create a new string $S = X_\$ \, \#_1 \, X_\$^R \, \#_2$ over the alphabet $\Sigma^{\$\#}$ and preprocess S to construct its suffix tree, so that we may perform LCE queries on S in $\mathcal{O}(1)$ time.

In the process of constructing $X_\$$ and S, we also build a matching table $\mathcal{M}$ over the alphabet $\Sigma^{\$\#}$. We define the matching table $\mathcal{M} : \Sigma^{\$\#} \times \Sigma^{\$\#} \to \{\texttt{true}, \texttt{false}\}$ by taking:

$$\mathcal{M}(s_1, s_2) = \begin{cases} s_1 = s_2 & \text{for } s_1, s_2 \in \Sigma \\ \tilde{X}[\texttt{loc}(s_1)] \approx \tilde{X}[\texttt{loc}(s_2)] & \text{for } s_1, s_2 \in \{\$_0, ..., \$_{k-1}\} \\ \tilde{X}[\texttt{loc}(s_1)] \approx s_2 & \text{for } s_1 \in \{\$_0, ..., \$_{k-1}\}, s_2 \in \Sigma \\ \tilde{X}[\texttt{loc}(s_2)] \approx s_1 & \text{for } s_1 \in \Sigma, s_2 \in \{\$_0, ..., \$_{k-1}\} \\ s_1 = s_2 & \text{for } s_1 \in \{\#_1, \#_2\} \vee s_2 \in \{\#_1, \#_2\} \end{cases}$$

for all $s_1, s_2 \in \Sigma^{\$\#}$. Note that it trivially follows from the commutativity of operations in the definition, that $\mathcal{M}(s_1, s_2) = \mathcal{M}(s_2, s_1)$ for any pair of characters $s_1, s_2 \in \Sigma^{\$\#}$.

3.2. Determining maximal palindromes

We now have $X_\$$, S, $\mathcal{M}$ and the ability to perform LCE operations on S in constant time. This gives us the necessary information to systematically find the maximal palindrome at each centre of $\tilde{X}$. This is done via repeated LCE queries on S. If we choose our LCE queries strategically, we may determine the radius of maximal palindromes of $\tilde{X}$ at each centre.

Lemma 2. *Let $\tilde{X}$ be a degenerate string of length n with matching table $\mathcal{M}$, and let $c \in \{0, \frac{1}{2}, 1, \frac{3}{2}, \dots, n-1\}$. Then $P = \tilde{X}[i\,..\,j]$ is a* maximal palindrome *of $\tilde{X}$ if and only if $i = \lceil c \rceil - e$ and $j = \lfloor c \rfloor + e$, where*

$$e = LCE(S, \lceil c \rceil, 2n - \lfloor c \rfloor, \mathcal{M})$$

and $S = X_\$ \, \#_1 \, X_\$^R \, \#_2$.

Proof. The right-to-left direction follows immediately from the definition of the LCE and the structure of S. For the left-to-right direction, let $P = \tilde{X}[i\,..\,j]$ be a maximal palindrome of $\tilde{X}$. We split the problem into two separate cases for even length and odd length palindromes.

- For an even length palindrome, the centre $c = c' + \frac{1}{2}$ for some $c' \in \{0, \dots, n-2\}$. We calculate e, the length of the longest pair of matching factors within X starting from index $\lceil c \rceil$ reading to the right and from index $\lfloor c \rfloor$ reading to the left. Since $S = X_\$ \, \#_1 \, X_\$^R \, \#_2$ is of length $2n+2$, we express this as $e = LCE(S, c + \frac{1}{2}, 2n - c + \frac{1}{2}, \mathcal{M})$. The maximal palindrome at centre c then corresponds to $P = \tilde{X}[c + \frac{1}{2} - e\,..\,c - \frac{1}{2} + e]$ if $e > 0$ or $P = \epsilon$ if $e = 0$.
- For an odd length palindrome, the centre $c \in \{0, \dots, n-1\}$. We calculate e, the length of the longest pair of matching factors within X starting from index $\lceil c \rceil (= c)$ reading to the right and from index $\lfloor c \rfloor (= c)$ reading to the left. Since $S = X_\$ \, \#_1 \, X_\$^R \, \#_2$ is of length $2n+2$, we express this as $e = LCE(S, c, 2n - c, \mathcal{M})$. The maximal palindrome at centre c then corresponds to $P = \tilde{X}[c - e\,..\,c + e]$.

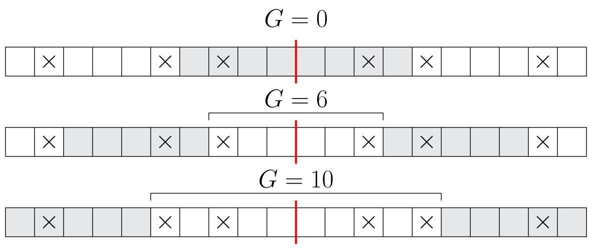


Fig. 5. Inverted repeats in a sequence for a given centre with 1 permitted mismatch. The centre is marked in red. The size of the gap is given by G. Mismatching symbols are marked with the symbol $\times$. The inverted repeat is indicated by the shaded cells. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

We may combine both the results for even and odd length palindromes into the statement of the lemma, through use of ceiling and floor notation. □

Thus we have a systematic way to determine the set of $2n - 1$ maximal palindromes $MP(\tilde{X})$ for any degenerate string $\tilde{X}$ of length n.

Theorem 1. *Given a degenerate string $\tilde{X}$ of length n over the alphabet Σ and a natural number k representing an upper bound on the number of non-solid symbols in $\tilde{X}$, all maximal palindromes of $\tilde{X}$ may be found in $\mathcal{O}\big(k|\Sigma|(k + \log|\Sigma|) + kn\big)$ time and $\mathcal{O}\big(k(k + |\Sigma|) + n\big)$ space.*

Proof. We analyse the various steps of the algorithm as described in this section earlier. To obtain the solid equivalent $X_\$$ of $\tilde{X}$, we construct the string $S = X_\$ \, \#_1 \, X_\$^R \, \#_2$ and construct the suffix tree of S, each require $\mathcal{O}(n)$ time and space.

It is reasonable to assume that non-solid symbols store their characters in lexicographical order. Under this assumption, determining a match between any two non-solid symbols is an $\mathcal{O}(|\Sigma|)$ operation, and between a non-solid and solid symbol is an $\mathcal{O}(\log(|\Sigma|))$ operation. It therefore follows that the preprocessing time to generate the portion of the matching table $\mathcal{M}$ comparing two symbols where at least one of which is a non-solid symbol is $\mathcal{O}\big(k|\Sigma|(k + \log|\Sigma|)\big)$. Since the portion of the matching table comparing characters $s_1, s_2 \in \Sigma$ is already known, no further preprocessing is required. Indeed, the required additional storage space to implicitly store the matching table $\mathcal{M}$ is thus $\mathcal{O}\big(k(k + |\Sigma|)\big)$, since we require $\mathcal{O}(k^2)$ space for storing all match results for pairs of non-solid symbols and $\mathcal{O}(k|\Sigma|)$ to store all match results for pairs of symbols consisting of a solid and a non-solid symbol.

Determining maximal palindromes requires us to consider each of the $2n - 1$ possible centres of $\tilde{X}$. For each of these centres, we perform a single $LCE(X, i, j, \mathcal{M})$ query. This is done by calculating $LCE_{k'}(X, i, j)$ for $k' \in [0, k]$, by observing the following:

$$LCE(X, i, j, \mathcal{M}) = \max(\{l_{k'} : \mathcal{M}(i + l_{k'-1} + 1, j + l_{k'-1} + 1) = \texttt{true}, k' \in [1, k]\} \cup \{l_0\})$$

where $l_{k'} = LCE_{k'}(X, i, j)$. Thus a single $LCE(X, i, j, \mathcal{M})$ query requires $\mathcal{O}(k)$ time by Lemma 1. Therefore we may determine the list of maximal palindromes $MP(\tilde{X})$ in $\mathcal{O}(kn)$ time and $\mathcal{O}(n)$ space, after initial preprocessing. □

3.3. Determining maximal palindromes with gaps and mismatches

We now extend this result to find degenerate palindromes with gaps and errors. For a given centre, the possible palindromes repeats are determined by first identifying symbols which are equidistant from the centre and are considered to mismatch using the kangaroo method.

The procedure next considers a minimal initial gap which is subsequently increased in order to reduce the number of mismatches inside the inverted repeat being considered, and thus permits a longer extension (see Fig. 5).

This demonstrates the principle of finding several unique palindromes with the same centre by extending the gap to effectively swallow an additional mismatch, such that the inverted repeat may be extended to the position directly adjacent to the next mismatch. This extending procedure is performed repeatedly to obtain all inverted repeats for a given centre, while taking into account the parameters specifying the maximum gap and the size range for the inverted repeat itself. The algorithm maintains efficiency by calculating only the necessary mismatch locations needed for a given set of parameters, and no more.

Theorem 2. *Given a degenerate string $\tilde{X}$ of length n, a gap size g, an upper bound on the number of mismatches permitted m over the alphabet Σ and a natural number k representing an upper bound on the number of non-solid symbols in $\tilde{X}$, all maximal palindromes of X may be found in $\mathcal{O}(k|\Sigma|(k + \log|\Sigma|) + (k + g + m)n)$ time and $\mathcal{O}(k(k + |\Sigma|) + (g + m)n)$ space.*

Proof. The preprocessing of the string $\tilde{X}$ is exactly same as described in Theorem 1, and requires $\mathcal{O}(k|\Sigma|(k+\log|\Sigma|)$ time and $\mathcal{O}\big(k(k+|\Sigma|)\big)$ space. After preprocessing, we consider the maximal palindromes for each of the $2n-1$ possible centres of $\tilde{X}$.

For each centre, we first identify all inverted repeats with a gap of size 0 and $\leq m$ mismatches. We perform an $LCE_m(X,i,j,\mathcal{M})$ query, which is done by calculating $LCE_{k'}(X,i_m,j_m)$ for $k' \in [0,k]$ m times, where $i_m = i_{m-1}+l_k+2$ and $j = j_{m-1}+l_k+2$ and $i_1 = i$, $j_1 = j$. Since the string is bounded by k degenerate symbols in X, a single $LCE_m(X,i,j,\mathcal{M})$ query will require $\mathcal{O}(m+k)$ time by Lemma 1.

Next, we define inverted repeats, denoted by IR with gap sizes $g>0$ and m mismatches recursively. Let $g_r = g_{r-1}+2l_{k'}$ for any $r \geq 0$ denote the gap size of an IR centred at some c, and let $g_0 = 0$. For g_r, we first calculate $l'_g = LCE_{k'}(X, i+g_{r/2-1}, j+g_{r/2-1})$ and then check if $g_r \leq g$. If it holds, an additional $LCE_{k'}(X,i,j)$ query is performed, where each i and j of the next $LCE_{k'}(X,i,j)$ are two positions directly after the end of the last $LCE_{k'}(X,i,j)$ query for g_{r-1}. Additionally, we remove the substring corresponding to l'_g and report the resulting maximal gapped inverted repeat.

Thus after preprocessing, we can determine all maximal palindromes with g gaps and m mismatches in $\mathcal{O}(k+g+m)n$ time and $\mathcal{O}(g+m)n$ space. □

3.4. Palindromic factorisation

It should also be briefly mentioned that the algorithms described here can also be applied to determine a palindromic factorisation of a given degenerate string. Palindromic factorisation is the process of splitting a string into adjacent factors such that all factors are palindromes and the minimal number of factors possible is used. We may add the further requirement that all palindromes in the factorisation are maximal palindromes, however, as highlighted earlier, this may not always be possible. This is a theoretical result with no known direct application to biological sequences, but is a byproduct of the main algorithm. Note that it is possible to have multiple valid factorisations of a string $\tilde{X}$ which contain the same number of palindromic factors. Alternatively, there may be no valid maximal palindromic factorisation of $\tilde{X}$, which differs to ordinary palindromic factorisation. This is a result of the fact that a single character is guaranteed to be an ordinary palindrome, but not necessarily a maximal palindrome.

Degenerate Maximal Palindromic Factorisation
Input: Degenerate string $\tilde{X}$ of length n and a natural number k representing an upper bound on the number of non-solid symbols in $\tilde{X}$.
Output: Array of indexes $T_0, T_1, \ldots, T_m$ describing a maximal palindromic factorisation $f_1, f_2, \ldots, f_m$ where $f_i = \tilde{X}[T_{i-1} \mathinner{..} T_i - 1]$ is a maximal palindrome for all $i \in \{1,\ldots,m\}$ and the number of factors m is minimised.

Lemma 3. *Given a string $\tilde{X}$ of length n containing k degenerate symbols, the factorisation of $\tilde{X}$, where each factor is a maximal palindrome with a gap of up to g symbols and up to m mismatches, can be done in $\mathcal{O}\big(k|\Sigma|(k+\log|\Sigma|)+(k+g+m)n\big)$ time and $\mathcal{O}\big(k(k+|\Sigma|)+(g+m)n\big)$ space.*

Proof. From the result of Theorem 2, we have established that identifying all of the maximal palindromes with up to g gaps and m mismatches are done in $\mathcal{O}\big(k|\Sigma|(k+\log|\Sigma|)+(k+g+m)n\big)$ time and $\mathcal{O}\big(k(k+|\Sigma|)+(g+m)n\big)$ space. Following this, we now construct a graph, where indices of the string $\tilde{X}$ correspond to vertices and maximal palindromes correspond to edges. Specifically, we build a graph G with vertices V and edges E, where $V = \{0,\ldots,n\}$ and $E = \{(i,j+1) : (i,j) \in MP(\tilde{X})\}$. The construction of the graph G requires $\mathcal{O}(g+m)n$ time and $\mathcal{O}(g+m)n$ space. Performing a single breadth first search on G is a standard $\mathcal{O}(g+m)n$ time operation since the number of edges $|E|$ and number of vertices $|V|$ are bounded by $\mathcal{O}(g+m)n$. Thus by considering the total complexity of these individual steps, the final complexity of the algorithm follows. □

The corresponding pseudocode to our algorithms are described in Algorithms 1-3. The entry point of the pseudocode is the function getFactorisation. This accepts an array $\tilde{X}$ of length n where every element in the array is a string representing the possible characters of a degenerate symbol. A string of length 1 is thus a solid symbol. The input parameter k specifies an upper bound on the number of non-solid symbols in $\tilde{X}$. The list of maximal palindromes for each centre is returned. We make use of some functions for which the pseudocode is not given. We detail those functions here:

LCE(X,i,j) Longest common extension function. Given a solid string X returns the length of the longest common prefix between the ith and jth suffix of X. Open-source implementations are available using suffix trees or suffix arrays. Details are given in the Preliminary section.

getMatchTable$(\tilde{X},n,k)$ Given a degenerate string $\tilde{X}$ of length n over the alphabet Σ and an upper bound on non-solid symbols k, returns the matching table $\mathcal{M}$ over the alphabet $\Sigma^{\$\#}$ as described in the Preprocessing subsection. The specific implementation of this function and the returned matching table data structure is dependent on the data structures used to implement degenerate strings. Given any two symbols s_1 and s_2 from the set $\Sigma^{\$\#}$, $\mathcal{M}(s_1, s_2)$ gives a `true` or `false` value indicating whether or not s_1 matches s_2.

getShortestPath$(G, \mathit{source}, \mathit{target})$ Given a directed graph G of vertices and edges, returns an array of integers containing the sequence of vertex labels encountered on the shortest path from the *source* vertex to the *target* vertex. If no possible path exists it returns `null`. If multiple shortest paths exist, the chosen path will depend on the specifics of the implementation. For example the array $[0, 2, 7, 10]$ is returned if the shortest path from *source* vertex 0 to *target* vertex 10 is via vertices 2 and 7. The shortest path can be found by performing a breadth first search on G starting at *source* and choosing the path that first encounters the *target* during the search.

Algorithm 1 Degenerate Maximal Palindromes.

```
function getMaximalPalindromes($\tilde{X}, n, k$)
    $X_\$$ = empty string
    $j = 0$
    for $i = 0$ to $n - 1$ do                    ▷ build string with non-solid symbols substituted with λ values
        $char = \tilde{X}[i]$
        if length($char$) == 1 then
            $X_\$ = X_\$ + char$                 ▷ operator + on strings is concatenation
        else
            $X_\$ = X_\$ + \$_j$
            $j = j + 1$
        end if
    end for
    $\mathcal{M}$ = getMatchTable($\tilde{X}, n, k$)
    $S = X_\$ + \#_1 +$ reverse($X_\$$) $+ \#_2$   ▷ string used to find palindromes
    return $even\_palindromes$ = getPalindromes($S, n, k, 0, \mathcal{M}$)
    return $odd\_palindromes$ = getPalindromes($S, n, k, 1, \mathcal{M}$)

    $G$ = directed graph with vertices 0 to $n$

    for $p \in even\_palindromes \cup odd\_palindromes$ do   ▷ build graph of all palindromes
        if $p$ is not null then
            add edge ($p.left$, $p.right + 1$) to $G$
        end if
    end for

    return getShortestPath($G, 0, n$)
end function
```

Algorithm 2 Degenerate Maximal Palindromes.

```
function getPalindromes($S, n, k, isOdd, \mathcal{M}$)             ▷ $isOdd$ is 0 or 1
    $palindromes$ = array of length ($n - 1 + isOdd$) of coordinates (0, 0)
    for $i = 1$ to ($n - 1 + isOdd$) do
        $j = 2n - i + 1 + isOdd$
        $e$ = realLCE($S, 2n + 2, k, i, j, \mathcal{M}$)       ▷ determine maximal extension from centre
        $left = i - e - isOdd$
        $right = i + e - 1$
        if $left \le right$ then                            ▷ store maximal palindrome if found
            $palindromes[i - 1] = (left, right)$
        else
            $palindromes[i - 1]$ = null
        end if
    end for
    return $palindromes$                                  ▷ return list of maximal palindromes found
end function
```

4. Experimental results

The following complete genomes of SARS-CoV-2 were downloaded from ViPR [43] as representative strains for highlighted Lineages of Concern: MW64206.1, MW598408.1, MZ184193.1, MZ315637.1, MZ436591.1, MZ358404.1, MZ505747.1 and MW642248.1. Each genome was analyzed for IRs with halves of length 4-1000 nucleotides (nt); 0, ≤ 1, and ≤ 2 mismatches; and varying gap sizes up to 100 nt. We experimentally quantify the number of IRs in each genome; the complete tables of IR frequencies are listed in Appendix A, where the results have been organised by IR-half length and exact number of mismatches. Fig. 6 shows the average distribution of IRs with half length ℓ across each lineage for varying input parameters of $\le m$ mismatches and $\le g$ gaps permitted.

Algorithm 3 Degenerate Maximal Palindromes.

```
function REALLCE($S, n, k, i, j, \mathcal{M}$)
    $real\_lce = 0$
    $mismatch\_count = 0$
    while $mismatch\_count < k + 1$ do
        $real\_lce = real\_lce + \text{LCE}(S, i + real\_lce, j + real\_lce)$
        if $i + real\_lce \geq n$ or $j + real\_lce \geq n$ then
            break
        end if
        $s_1 = S[i + real\_lce]$
        $s_2 = S[j + real\_lce]$
        if $\mathcal{M}(s_1, s_2)$ then
            $real\_lce = real\_lce + 1$
        else
            break
        end if
        $mismatch\_count = mismatch\_count + 1$
    end while
    return $real\_lce$
end function
```

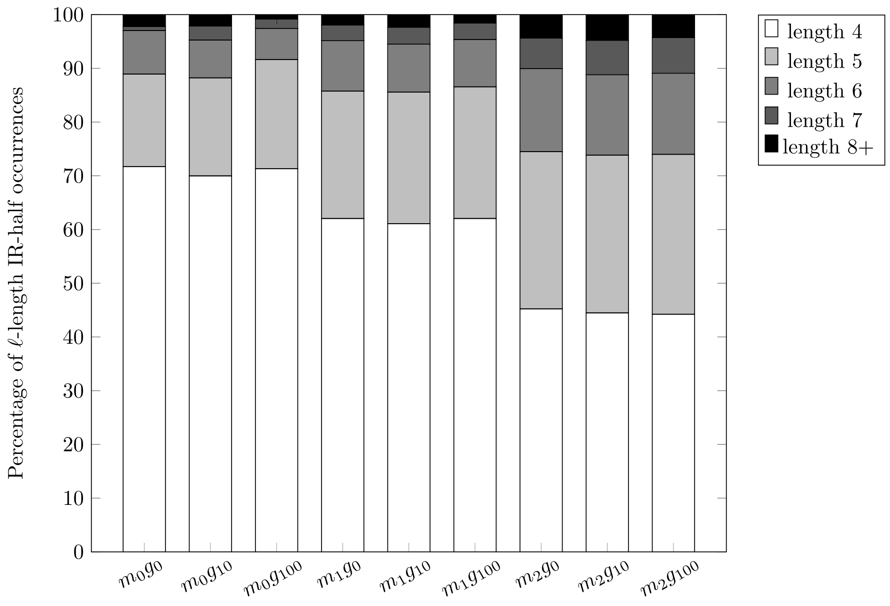


Fig. 6. Percentage of IRs with half length ℓ with $\leq m$ permitted mismatches and $\leq g$ permitted gap size.

Across every set of parameters, we see that the majority (over 50%) of the IRs have half lengths of 4, 5 and 6, whereas longer IRs become considerably less common as the length increases. It appears that increasing the gap does not largely change the distribution of inverted repeats, however increasing the number of mismatches results in an increased percentage of IRs with a half length of ≥ 5. Note that not all IRs are guaranteed to increase when the number of errors increase since they are maximal (i.e. cannot be extended to the left or right unless further mismatches have been utilised). The leftmost and rightmost symbol in any reported IR must necessarily match. One interesting observation was that when the maximum number of mismatches permitted increased to 2 (for gap sizes ≤ 0, ≤ 10 and ≤ 100), there appears to be a single occurrence of an IR with half length 18, however no occurrences of IRs with half length 16 or 17. This pattern was observed across each of the strains tested.

The complete table of standard deviations of ℓ length IRs across each strain for varying input parameters of $\leq m$ mismatches and $\leq g$ gap size is listed in Appendix B.

For a given inverted repeat with half length ℓ, the standard deviation was observed to generally increase with respect to the permitted gap size and/or number of mismatches increasing. For a fixed gap size and fixed maximum number of mismatches, the standard deviation was observed to generally decrease the longer the IR is, which is most likely attributed to the frequency of inverted repeats with half length ℓ decreasing as ℓ increases. Inverted Repeats with half length 12 were an exception to this (occurring in configurations m_1g_{10}, m_1g_{100}, m_2g_0, m_2g_{10} and m_2g_{100}). In such cases, the standard deviation across all strains was consistently lower compared to the standard deviation for IRs with half length 13.

Table 2
Mean and standard deviation of the total number of IRs located with up to m mismatches and g maximum gap length.

Number of Mismatches	Maximum Gap length	μ (σ)
0	0	134.38 (1.30)
0	10	1451.75 (2.96)
0	100	12001.13 (40.61)
1	0	891.38 (1.30)
1	10	9003.88 (7.95)
1	100	78210.13 (66.86)
2	0	2336.50 (2.45)
2	10	22907.13 (17.00)
2	100	195934.75 (189.63)

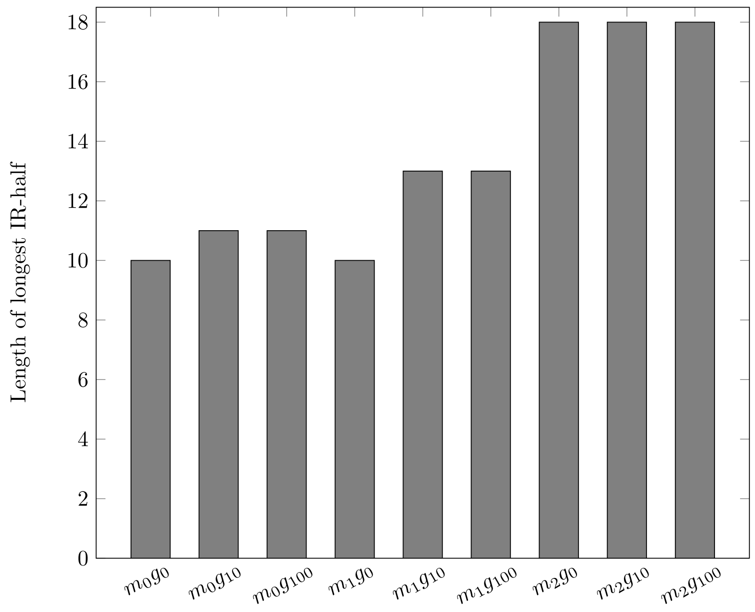


Fig. 7. Length of longest IR halves detected across all eight strains with m mismatches and g gaps.

Table 2 shows the mean (μ) and standard deviation (σ) across the total number of IRs located per variant. Here we can see that increasing the gap and/or number of mismatches always results in an increased number of IR detected. In each case, the deviation between strains is extremely small – always less than 1%.

4.1. Longest inverted repeats

Long inverted repeats are often associated with influencing genome stability within various organisms. Fig. 7 shows the trend of the longest inverted repeats for gap size g and maximum number of mismatches m. These values were the same across each variant, though there was some variance with the frequency of IRs found with these lengths.

The longest palindrome (i.e. inverted repeat with gap size zero) was the 20 nt sequence ACACTGGTAATTACCAGTGT. The effect of increasing the gap size shows to increase the overall length of the IR-half, and has the largest effect when the maximum number of mismatches was 1. When the permitted gap size was increased up to 100, the longest IR-half with no mismatches was the 11 nt sequence AGGTAAAACAT...ATGTTTTACCT.

Increasing the number of permitted mismatches also resulted in an increase of the overall length of the longest IR-half, particularly when this changed from 1 to 2 mismatches. When the number of allowed mismatches was increased to 2 (whilst retaining a gap size of zero), it can be observed that the length of the longest IR was the 36 nt (i.e. half length 18) sequence TAGTGAGTACACTGGTAATTACCAGTGTGGTCACTA. Notably, when both the permitted gap length and number of mismatches were maximised to ≤ 100 and ≤ 2 respectively, the same 36 nt sequence remained the longest maximal IR identified.

5. Discussion and conclusion

Inverted repeats, nucleotide sequences followed by their reverse complement, serve diverse purposes and can vary in length. In our study, we have presented efficient algorithms to identify inverted repeats with permitted gaps and mismatches, and have demonstrated our algorithm by quantifying inverted repeats on a set of SARS-CoV-2 genomes. From our experimental results, we identified an IR with half length 18 across all strains when the number of mismatches was increased to 2, however there was a lack of half length IRs of length 16 and 17. Furthermore, we also observed that the standard deviation of IRs with half length 12 was lower than expected. Performing an analysis of the secondary structure of the genome may reveal further significance of the patterns and how they contribute to the genomic signature of SARS-CoV-2.

As future work, it may be of interest to investigate applications of our algorithm to computationally improve existing methods for the prediction of secondary structures from genomic strings. Current methods that directly parse a sequence typically involve exploring the conformational space available to the RNA and categorising the computed structures, however the number of possible secondary structure combinations scales exponentially, which is not ideal when analysing larger genomes [44]. We anticipate that identifying common genomic string patterns including inverted repeats will prune down the search space considerably, thus improving the overall space and time.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Frequency of inverted repeats

In this appendix, we present the frequency of maximal Inverted Repeats (IRs) with half length ℓ occurring in MW642026.1, MW598408.1, MZ184193.1, NZ315637.1, MZ436591.1, MZ358404.1, MZ505747.1, MW642248.1 containing a gap with maximum size g and up to m permitted mismatches. The order of the arrays in the queue correspond to the number of IRs found with their length from 0 to 20. Each position within the array correlates to the number of IRs found with the same number of mismatches starting from 0. (See Tables A.3–A.10.)

Table A.3
MW642026.1

g	m	Frequency of Palindromes
0	0	[0], [0], [0], [0], [94], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [68, 488], [19, 190], [10, 72], [1, 25], [2, 8], [0, 3], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [51, 358, 649], [14, 145, 520], [9, 54, 297], [1, 17, 117], [1, 6, 56], [0, 2, 18], [0, 1, 8], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1016], [264], [101], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [632, 4866], [164, 2036], [55, 747], [19, 263], [15, 119], [3, 41], [2, 27], [1, 3], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [416, 3069, 6717], [103, 1304, 5323], [39, 463, 2911], [10, 148, 1311], [7, 74, 543], [3, 26, 221], [0, 11, 118], [1, 2, 43], [0, 2, 28], [0, 2, 12], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8599], [2438], [695], [205], [69], [24], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4732, 43781], [1338, 17843], [378, 6516], [106, 2261], [39, 809], [15, 240], [4, 116], [1, 29], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2623, 24296, 59813], [737, 9703, 47901], [227, 3625, 25635], [60, 1225, 11781], [18, 425, 4822], [9, 137, 1779], [1, 57, 735], [1, 17, 264], [0, 2, 104], [0, 4, 37], [0, 0, 9], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.4
MW598408.1

g	m	Frequency of Palindromes
0	0	[0], [0], [0], [0], [96], [23], [10], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [68, 486], [19, 193], [9, 74], [1, 26], [2, 8], [0, 3], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [51, 356, 648], [14, 148, 522], [8, 57, 297], [1, 17, 114], [1, 6, 55], [0, 2, 19], [0, 1, 10], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1018], [267], [99], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [628, 4872], [166, 2036], [55, 750], [19, 264], [15, 119], [3, 40], [2, 28], [1, 3], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [413, 3062, 6717], [106, 1306, 5333], [38, 465, 2922], [10, 145, 1307], [7, 73, 539], [3, 25, 227], [0, 12, 122], [1, 2, 43], [0, 2, 28], [0, 2, 11], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8589], [2441], [695], [202], [74], [24], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]

Table A.4 (*continued*)

g	*m*	Frequency of Palindromes
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4730, 43841], [1339, 17831], [380, 6502], [105, 2278], [42, 801], [15, 235], [4, 120], [1, 29], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2631, 24302, 59825], [734, 9701, 47923], [226, 3614, 25701], [60, 1236, 11750], [19, 420, 4823], [9, 131, 1798], [1, 61, 733], [1, 18, 264], [0, 2, 104], [0, 4, 37], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.5
MW642248.1

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [96], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [69, 489], [19, 191], [10, 72], [1, 24], [2, 8], [0, 3], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [52, 360, 646], [14, 146, 523], [9, 54, 297], [1, 15, 118], [1, 6, 53], [0, 2, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1021], [262], [102], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [634, 4890], [163, 2031], [57, 743], [19, 259], [15, 119], [3, 40], [2, 27], [1, 4], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [415, 3074, 6702], [104, 1301, 5342], [39, 458, 2922], [10, 143, 1317], [7, 75, 538], [3, 25, 225], [0, 11, 119], [1, 3, 41], [0, 2, 28], [0, 2, 11], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8618], [2438], [692], [205], [73], [22], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4751, 43866], [1347, 17811], [377, 6496], [107, 2272], [41, 793], [14, 237], [4, 121], [1, 31], [0, 4], [0, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2634, 24287, 59749], [739, 9696, 47911], [226, 3610, 25693], [61, 1232, 11772], [19, 415, 4821], [8, 131, 1808], [1, 63, 729], [1, 19, 266], [0, 2, 103], [0, 3, 38], [0, 0, 8], [0, 0, 4], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.6
MZ184193.1

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [97], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [69, 482], [19, 191], [10, 75], [1, 26], [2, 8], [0, 3], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [52, 354, 651], [14, 147, 522], [9, 57, 294], [1, 17, 115], [1, 6, 54], [0, 2, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1020], [264], [102], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [630, 4868], [166, 2039], [58, 747], [19, 263], [15, 118], [3, 39], [2, 27], [1, 3], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [414, 3062, 6722], [104, 1306, 5311], [40, 458, 2927], [10, 147, 1314], [7, 74, 541], [3, 24, 225], [0, 11, 126], [1, 2, 42], [0, 2, 28], [0, 2, 10], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8582], [2443], [689], [207], [72], [24], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4718, 43856], [1344, 17855], [376, 6505], [107, 2271], [41, 805], [15, 242], [4, 118], [1, 28], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2614, 24293, 59838], [738, 9713, 47879], [225, 3610, 25730], [58, 1228, 11766], [19, 418, 4816], [9, 138, 1796], [1, 60, 743], [1, 17, 273], [0, 2, 104], [0, 4, 35], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.7
MZ315637.1

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [97], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [70, 482], [19, 193], [10, 73], [1, 24], [2, 8], [0, 4], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [53, 353, 650], [14, 149, 524], [9, 55, 297], [1, 15, 116], [1, 6, 54], [0, 3, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1016], [264], [103], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]

(*continued on next page*)

Table A.7 (*continued*)

g	*m*	Frequency of Palindromes
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [629, 4858], [166, 2040], [59, 746], [19, 262], [15, 118], [3, 39], [2, 27], [1, 3], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [412, 3052, 6710], [106, 1305, 5295], [40, 459, 2925], [10, 143, 1311], [7, 74, 541], [3, 24, 229], [0, 11, 125], [1, 2, 41], [0, 2, 28], [0, 2, 10], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8526], [2433], [694], [209], [71], [23], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4673, 43779], [1342, 17799], [380, 6490], [107, 2257], [41, 806], [14, 243], [4, 117], [1, 27], [0, 4], [0, 5], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2632, 24251, 59677], [736, 9681, 47762], [226, 3598, 25699], [57, 1215, 11653], [19, 418, 4797], [8, 139, 1793], [1, 59, 744], [1, 17, 273], [0, 2, 104], [0, 5, 34], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.8
MZ436591.1

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [97], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [69, 482], [19, 193], [10, 75], [1, 25], [2, 8], [0, 3], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [52, 354, 649], [14, 149, 522], [9, 57, 293], [1, 16, 115], [1, 6, 54], [0, 2, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[[0], [0], [0], [0], [1016], [263], [102], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [629, 4867], [164, 2034], [58, 748], [19, 263], [15, 119], [3, 40], [2, 26], [1, 3], [0, 3], [0, 2], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [414, 3059, 6717], [104, 1302, 5313], [40, 460, 2925], [10, 144, 1310], [7, 75, 540], [3, 25, 227], [0, 10, 124], [1, 2, 42], [0, 2, 28], [0, 2, 10], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8511], [2443], [689], [207], [72], [25], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4659, 43880], [1342, 17805], [373, 6489], [107, 2270], [41, 806], [16, 244], [4, 119], [1, 29], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2621, 24316, 59721], [737, 9683, 47831], [224, 3609, 25737], [57, 1222, 11664], [19, 417, 4777], [10, 137, 1794], [1, 60, 747], [1, 18, 274], [0, 2, 105], [0, 4, 35], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.9
MZ358404.1.

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [97], [23], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [70, 482], [19, 193], [10, 75], [1, 25], [2, 8], [0, 4], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [53, 354, 649], [14, 148, 523], [9, 57, 296], [1, 16, 115], [1, 6, 54], [0, 3, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1015], [267], [103], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [628, 4877], [168, 2040], [58, 743], [19, 265], [15, 117], [4, 40], [2, 27], [1, 3], [0, 3], [0, 1], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [412, 3060, 6708], [105, 1306, 5307], [40, 457, 2936], [10, 145, 1307], [7, 73, 538], [3, 25, 231], [0, 11, 126], [1, 2, 42], [0, 2, 28], [0, 1, 11], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8519], [2440], [694], [210], [71], [24], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4664, 43842], [1343, 17820], [381, 6489], [107, 2271], [41, 801], [16, 245], [4, 116], [1, 27], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2628, 24295, 59754], [734, 9679, 47826], [228, 3595, 25729], [57, 1227, 11642], [19, 415, 4804], [9, 141, 1798], [1, 58, 740], [1, 17, 273], [0, 2, 104], [0, 4, 35], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Table A.10
MZ505747.1.

g	*m*	Frequency of Palindromes
0	0	[0], [0], [0], [0], [97], [24], [11], [1], [2], [0], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]
0	1	[0, 0], [0, 0], [0, 0], [0, 0], [70, 481], [20, 193], [10, 74], [1, 25], [2, 8], [0, 4], [1, 3], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]

Table A.10 (*continued*)

g	m	Frequency of Palindromes
0	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [53, 353, 651], [15, 148, 525], [9, 56, 295], [1, 16, 115], [1, 6, 54], [0, 3, 19], [0, 1, 9], [0, 0, 4], [0, 0, 3], [0, 0, 1], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
10	0	[0], [0], [0], [0], [1006], [268], [103], [38], [22], [4], [4], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
10	1	[0, 0], [0, 0], [0, 0], [0, 0], [624, 4877], [168, 2045], [58, 742], [19, 264], [15, 119], [4, 41], [2, 27], [1, 3], [0, 3], [0, 1], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
10	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [411, 3059, 6710], [106, 1309, 5314], [40, 455, 2930], [10, 144, 1309], [7, 75, 538], [3, 26, 232], [0, 11, 124], [1, 2, 42], [0, 2, 28], [0, 1, 11], [0, 0, 2], [0, 0, 2], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]
100	0	[0], [0], [0], [0], [8521], [2440], [698], [211], [70], [24], [7], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]
100	1	[0, 0], [0, 0], [0, 0], [0, 0], [4663, 43830], [1340, 17815], [383, 6501], [108, 2273], [41, 804], [16, 246], [4, 115], [1, 28], [0, 4], [0, 4], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]
100	2	[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [2623, 24277, 59747], [733, 9675, 47838], [228, 3594, 25710], [57, 1229, 11648], [19, 416, 4809], [9, 142, 1800], [1, 57, 750], [1, 18, 271], [0, 2, 103], [0, 4, 35], [0, 0, 8], [0, 0, 5], [0, 0, 0], [0, 0, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]

Appendix B. Standard deviation

In this appendix we present the Standard Deviation (3 d.p.) of maximal IRs with half length ℓ across MW642026.1, MW598408.1, MZ184193.1, NZ315637.1, MZ436591.1, MZ358404.1, MZ505747.1, MW642248.1 for a maximum number of mismatches, $m \leq 0, 1, 2$ and maximum gap size, $g \leq 0, \leq 10, \leq 100$. Cells containing dashes (-) indicate no IRs with half length ℓ were identified.

ℓ	m_0g_0	m_0g_{10}	m_0g_{100}	m_1g_0	m_1g_{10}	m_1g_{100}	m_2g_0	m_2g_{10}	m_2g_{100}
4	1.061	4.569	42.997	2.642	10.589	52.789	1.488	9.620	64.464
5	0.354	2.167	3.251	1.389	6.018	18.843	2.766	13.933	65.713
6	0.354	1.356	3.105	1.302	2.507	10.128	1.069	5.731	26.452
7	0	0	2.976	0.756	1.808	6.728	1.165	3.739	65.006
8	0	0	1.604	0.518	0.756	4.536	0.886	1.959	17.096
9	0	0	0.886	0.268	1.035	4.438	0.707	3.603	8.035
10	0	0	0	0	0.535	2.121	0.535	3.024	6.347
11	-	0	0	-	0.354	1.309	0	0.641	4.138
12	-	-	-	-	0	0	0	0	0.641
13	-	-	-	-	0.463	0.535	0	0.756	1.035
14	-	-	-	-	-	-	-	0	0.354
15	-	-	-	-	-	-	0	0	0.354
16	-	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	0	0	0
19	-	-	-	-	-	-	-	-	-
20	-	-	-	-	-	-	-	-	-

References

[1] I. Voineagu, V. Narayanan, K.S. Lobachev, S.M. Mirkin, Replication stalling at unstable inverted repeats: interplay between DNA hairpins and fork stabilizing proteins, Proc. Natl. Acad. Sci. USA 105 (29) (2008) 9936–9941, https://doi.org/10.1073/pnas.0804510105.

[2] C. Pearson, H. Zorbas, G.B. Price, M. Zannis-Hadjopoulos, Inverted repeats, stem-loops, and cruciforms: significance for initiation of DNA replication, J. Cell. Biochem. 63 (1) (1996) 1–22, https://doi.org/10.1002/(SICI)1097-4644(199610)63:1<1::AID-JCB1>3.0.CO;2-3.

[3] J.J. Bissler, DNA inverted repeats and human disease, Front. Biosci. 3 (1998) 408–418, https://doi.org/10.2741/a284.

[4] D. Cain, O. Erlwein, A. Grigg, R.A. Russell, M.O. McClure, Palindromic sequence plays a critical role in human foamy virus dimerization, J. Virol. 75 (8) (2001) 3731–3739, https://doi.org/10.1128/JVI.75.8.3731-3739.2001.

[5] A.M.G. Dirac, H. Huthoff, J. Kjems, B. Berkhout, Requirements for RNA heterodimerization of the human immunodeficiency virus type 1 (hiv-1) and hiv-2 genomes, J. Gen. Virol. 83 (10) (2002) 2533–2542, https://doi.org/10.1099/0022-1317-83-10-2533.

[6] M.K. Hill, M. Shehu-Xhilaga, S.M. Campbell, P. Poumbourios, S.M. Crowe, J. Mak, The dimer initiation sequence stem-loop of human immunodeficiency virus type 1 is dispensable for viral replication in peripheral blood mononuclear cells, J. Virol. 77 (15) (2003) 8329–8335, https://doi.org/10.1128/jvi.77.15.8329-8335.2003.

[7] S. Karlin, C. Burge, A.M. Campbell, Statistical analyses of counts and distributions of restriction sites in DNA sequences, Nucleic Acids Res. 20 (6) (1992) 1363–1370, https://doi.org/10.1093/nar/20.6.1363.

[8] M.-Y. Leung, K.P. Choi, A. Xia, L.H.Y. Chen, Nonrandom clusters of palindromes in herpesvirus genomes, J. Comput. Biol. 12 (3) (2005) 331–354, https://doi.org/10.1089/cmb.2005.12.331.

[9] E.P. Rocha, A. Danchin, A. Viari, Evolutionary role of restriction/modification systems as revealed by comparative genome analysis, Genome Res. 11 (2001) 946–958, https://doi.org/10.1101/gr.153101.

[10] D.S.H. Chew, K.P. Choi, H. Heidner, M.-Y. Leung, Palindromes in Sars and other coronaviruses, INFORMS J. Comput. 16 (4) (2004) 331–340, https://doi.org/10.1287/ijoc.1040.0087.

[11] R. Merkl, H.J. Fritz, Statistical evidence for a biochemical pathway of natural, sequence-targeted G/C to C/G transversion mutagenesis in Haemophilus influenzae Rd, Nucleic Acids Res. 24 (21) (1996) 4146–4151, https://doi.org/10.1093/nar/24.21.4146.

[12] E.P.C. Rocha, A. Viari, A. Danchin, Oligonucleotide bias in bacillus subtilis: general trends and taxonomic comparisons, Nucleic Acids Res. 26 (12) (1998) 2971–2980, https://doi.org/10.1093/nar/26.12.2971.

[13] Y. Almirantis, P. Charalampopoulos, J. Gao, C.S. Iliopoulos, M. Mohamed, S.P. Pissis, D. Polychronopoulos, On avoided words, absent words, and their application to biological sequence analysis, Algorithms Mol. Biol. 12 (1) (2017) 5, https://doi.org/10.1186/s13015-017-0094-z.

[14] G. Manacher, A new linear-time “on-line” algorithm for finding the smallest initial palindrome of a string, J. ACM 22 (3) (1975) 346–351, https://doi.org/10.1145/321892.321896.

[15] A. Apostolico, D. Breslauer, Z. Galil, Parallel detection of all palindromes in a string, Theor. Comput. Sci. 141 (1) (1995) 163–173, https://doi.org/10.1016/0304-3975(94)00083-U.

[16] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.

[17] S. Gupta, R. Prasad, S. Yadav, Searching gapped palindromes in DNA sequences using dynamic suffix array, Indian J. Sci. Technol. 8 (23) (2015) 1–9, https://doi.org/10.17485/ijst/2015/v8i23/70645.

[18] Y. Fujishige, M. Nakamura, S. Inenaga, H. Bannai, M. Takeda, Finding gapped palindromes online, in: International Workshop on Combinatorial Algorithms, Springer, 2016, pp. 191–202.

[19] R. Kolpakov, G. Kucherov, Searching for gapped palindromes, Theor. Comput. Sci. 410 (51) (2009) 5365–5373.

[20] G. Fici, T. Gagie, J. Kärkkäinen, D. Kempa, A subquadratic algorithm for minimum palindromic factorization, J. Discret. Algorithms 28 (2014) 41–48.

[21] T. I, S. Sugimoto, S. Inenaga, H. Bannai, M. Takeda, Computing palindromic factorizations and palindromic covers on-line, in: Symposium on Combinatorial Pattern Matching, Springer, 2014, pp. 150–161.

[22] A. Alatabbi, C.S. Iliopoulos, M.S. Rahman, Maximal palindromic factorization, in: Stringology, 2013, pp. 70–77.

[23] M. Rubinchik, A.M. Shur, Eertree: an efficient data structure for processing palindromes in strings, in: International Workshop on Combinatorial Algorithms, Springer, 2015, pp. 321–333.

[24] Z. Galil, J. Seiferas, A linear-time on-line recognition algorithm for“palstar”, J. ACM 25 (1) (1978) 102–111.

[25] D. Kosolobov, M. Rubinchik, A.M. Shur, Pal k is linear recognizable online, in: International Conference on Current Trends in Theory and Practice of Informatics, Springer, 2015, pp. 289–301.

[26] M. Adamczyk, M. Alzamel, P. Charalampopoulos, J. Radoszewski, Palindromic decompositions with gaps and errors, Int. J. Found. Comput. Sci. 29 (08) (2018) 1311–1329.

[27] M. Alzamel, J. Gao, C.S. Iliopoulos, C. Liu, S.P. Pissis, Efficient computation of palindromes in sequences with uncertainties, in: G. Boracchi, L.S. Iliadis, C. Jayne, A. Likas (Eds.), Engineering Applications of Neural Networks - 18th International Conference, Proceedings, EANN 2017, Athens, Greece, August 25-27, 2017, in: Communications in Computer and Information Science, vol. 744, Springer, 2017, pp. 620–629.

[28] K. Abrahamson, Generalized string matching, SIAM J. Comput. 16 (6) (1987) 1039–1051.

[29] M. Crochemore, C.S. Iliopoulos, T. Kociumaka, J. Radoszewski, W. Rytter, T. Waleń, Covering problems for partial words and for indeterminate strings, Theor. Comput. Sci. 698 (2017) 25–39.

[30] C.S. Iliopoulos, J. Radoszewski, Truly subquadratic-time extension queries and periodicity detection in strings with uncertainties, in: 27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2016, 8.

[31] N. Pisanti, H. Soldano, M. Carpentier, J. Pothier, A relational extension of the notion of motifs: application to the common 3D protein substructures searching problem, J. Comput. Biol. 16 (12) (2009) 1635–1660.

[32] H. Soldano, A. Viari, M. Champesme, Searching for flexible repeated patterns using a non-transitive similarity relation, Pattern Recognit. Lett. 16 (3) (1995) 233–246.

[33] C.S. Iliopoulos, R. Kundu, S.P. Pissis, Efficient pattern matching in elastic-degenerate strings, Inf. Comput. 279 (2021) 104616, https://doi.org/10.1016/j.ic.2020.104616.

[34] R. Grossi, C.S. Iliopoulos, C. Liu, N. Pisanti, S.P. Pissis, A. Retha, G. Rosone, F. Vayani, L. Versari, On-line pattern matching on similar texts, in: 28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 9.

[35] G. Bernardini, N. Pisanti, S.P. Pissis, G. Rosone, Pattern matching on elastic-degenerate text with errors, in: International Symposium on String Processing and Information Retrieval, Springer, 2017, pp. 74–90.

[36] G. Bernardini, P. Gawrychowski, N. Pisanti, S.P. Pissis, G. Rosone, Even faster elastic-degenerate string matching via fast matrix multiplication, in: C. Baier, I. Chatzigiannakis, P. Flocchini, S. Leonardi (Eds.), 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece, in: LIPIcs, vol. 132, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 21.

[37] A. Cisłak, S. Grabowski, J. Holub, Sopang: online text searching over a pan-genome, Bioinformatics 34 (24) (2018) 4290–4292.

[38] Computational Pan-Genomics: Status, Promises and Challenges, Brief. Bioinform. 19 (1) (2018) 118–135.

[39] Z. Galil, R. Giancarlo, Improved string matching with *k* mismatches, SIGACT News 17 (4) (1986) 52–54, https://doi.org/10.1145/8307.8309.

[40] G.M. Landau, U. Vishkin, Efficient string matching with *k* mismatches, Theor. Comput. Sci. 43 (1986) 239–249, https://doi.org/10.1016/0304-3975(86)90178-7.

[41] M. Farach, Optimal suffix tree construction with large alphabets, in: 38th Annual Symposium on Foundations of Computer Science, 1997, Proceedings, IEEE, 1997, pp. 137–143.

[42] M.A. Bender, M. Farach-Colton, The LCA Problem Revisited, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 88–94.

[43] B.E. Pickett, E.L. Sadat, Y. Zhang, J.M. Noronha, R.B. Squires, V. Hunt, M. Liu, S. Kumar, S. Zaremba, Z. Gu, et al., ViPR: an open bioinformatics database and analysis resource for virology research, Nucleic Acids Res. 40 (D1) (2012) D593–D598.

[44] J.T. Low, K.M. Weeks, Shape-directed RNA secondary structure prediction, Methods 52 (2) (2010) 150–158.