





Design of Worst-Case-Optimal Spaced Seeds

Jens Zentgraf   

Algorithmic Bioinformatics, Department of Computer Science, Saarland University,
Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science, Germany
Center for Bioinformatics Saar, Saarland Informatics Campus, Saarbrücken, Germany

Sven Rahmann   

Algorithmic Bioinformatics, Department of Computer Science, Saarland University,
Saarbrücken, Germany
Center for Bioinformatics Saar, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Read mapping (and alignment) is a fundamental problem in biological sequence analysis. For speed and computational efficiency, many popular read mappers tolerate only a few differences between the read and the corresponding part of the reference genome, which leads to reference bias: Reads with too many differences are not guaranteed to be mapped correctly or at all, because to even consider a genomic position, a sufficiently long *exact* match (*seed*) must exist.

While pangenomes and their graph-based representations provide one way to avoid reference bias by enlarging the reference, we explore an orthogonal approach and consider stronger substitution-tolerant primitives, namely *spaced seeds* or gapped k -mers. Given two integers $k \leq w$, one considers k selected positions, described by a *mask*, from each length- w window in a sequence. In the existing literature, masks with certain *probabilistic* guarantees have been designed for small values of k .

Here, for the first time, we take a combinatorial approach from a *worst-case* perspective. For any mask, using integer linear programs, we find least favorable distributions of sequence changes in two different senses: (1) minimizing the number of unchanged windows; (2) minimizing the number of positions covered by unchanged windows. Then, among all masks or all symmetric masks of a given shape (k, w) , we find the set of best masks that maximize these minima. As a result, we obtain robust masks, even for large numbers of changes.

We illustrate the properties of these masks by constructing a challenging set of reads that contain many approximately equidistributed substitutions (but no indels) that many existing tools cannot map, even though they are in principle easily mappable (apart from the large number of changes) because they originate from selected non-repetitive regions of the human reference genome. We observe that the majority of these reads can be mapped with a simple alignment-free approach using chosen spaced masks, where seeding approaches based on contiguous k -mers fail.

2012 ACM Subject Classification Applied computing → Molecular sequence analysis; Theory of computation → Discrete optimization; Applied computing → Bioinformatics

Keywords and phrases Spaced seed, Gapped k -mer, Integer linear program (ILP), Worst-case design, Reference bias

Digital Object Identifier 10.4230/LIPIcs.WABI.2025.22

Related Version

Previous Version: <https://www.biorxiv.org/content/10.1101/2023.11.20.56782>

Supplementary Material

Software (Source Code): <https://gitlab.com/rahmannlab/seed-optimization> [33]

archived at `swb:1:dir:81ca043ed372e91711c1a9255974224264b1eb5d`

Text (Supplementary tables): <https://doi.org/10.5281/zenodo.15690315>



© Jens Zentgraf and Sven Rahmann;

licensed under Creative Commons License CC-BY 4.0

25th International Conference on Algorithms for Bioinformatics (WABI 2025).

Editors: Broňa Brejová and Rob Patro; Article No. 22; pp. 22:1–22:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Read mapping and alignment are fundamental problems in biological sequence analysis required for many tasks in genomics and transcriptomics, like genome-wide variant calling, transcript expression quantification, or species identification and quantification in metagenomics, to name just a few. There exist efficient methods (in terms of both memory and running time) to map millions of reads against a reference genome, such as bwa-mem2 [39], bowtie2 [21], minimap2 [22, 23], strobealign [37], mapquik [9], BLEND [11] and many others.

Read mappers must balance efficiency against error-tolerance: Finding *exact* matches between reads and reference is extremely efficient with existing index data structures, and therefore error-tolerant index-based searches are reduced to exact searches of shorter fragments, using the pigeonhole principle or combinatorial search schemes [35]. Allowing an increasing number of differences (typically measured in terms of edit distance) between a read and the reference typically requires time that increases exponentially with the number of tolerated differences, because one has to evaluate many branching paths in an index data structure [24, 36]. Thus, for efficiency, the default settings of popular read mappers allow only a few errors, which leads to reference bias: Reads that differ too strongly from their best match in the reference genome may not be mapped at all, and the corresponding genomic variants may be missed during downstream variant calling steps [26].

One possible answer to the problem of reference bias has been the construction of pangenomes that represent not only a single linear sequence, but a large collection of possible genome sequences [6] of a species that can be compactly represented by a branching graph, such that known sequence variants are included in the pangenome. Consequently, read mapping algorithms and their underlying index data structures have been generalized to enable mapping linear reads against graph reference genomes [13, 34, 28]. Because many of the known variants are included in the reference, a small error threshold is now sufficient where a larger one was required before to map the same set of reads.

An orthogonal approach against reference bias is to design index data structures with built-in error tolerance. So far, this has proven easier for Hamming distance (substitutions only) than for edit distance (substitutions, insertions and deletions), because of the variability of the length of the match when indels are allowed. Spaced seeds, also called gapped k -mers, have been proposed and used as substitution-tolerant primitives: One considers k specifically selected positions, described by a *mask*, out of a larger window of $w \geq k$ positions. After their initial introduction by Burkhardt and Kärkkäinen [4], many studies have shown the superiority of spaced seeds (when used with appropriate masks) over standard k -mer seeds, e.g. for genome alignment [12], for homology detection [2, 43], for metagenomic classification [5], or for phylogenetic tree reconstruction [15]. In addition, these findings have spawned a large body of literature concerning designing *optimal* masks for given k and w and various specific objectives. In the following paragraphs, we attempt a brief (but necessarily incomplete) summary of the existing seed (or mask) design literature.

Lossless setting. Many articles focus on the *lossless* setting, which is related to but different from our focus, as will be explained below. In the lossless setting, the task is to find seeds (masks) for given k and w , a fixed number of changes c , and a fixed sequence length n , such that it can be guaranteed that we find an unchanged seed (a “hit”) in the sequence, independently of the positioning of the c changes among the n positions. The corresponding decision problem (“Is there an arrangement of the c changes such that there is no hit?”) has been shown to be NP-complete [29]. Algorithmically, the computations are based on

(exponential-time) dynamic programming algorithms over certain types of automata [3]. If already the decision problem for a fixed mask is hard, then choosing a best mask with given parameters, or a combination of masks, is certainly not easier, but asymptotic lower and upper bounds on parameters of optimal masks (with other parameters being fixed) have been obtained [10]. The problem becomes even more difficult and interesting when one considers multiple masks at once [20]. Recent work in uses heuristics based on periodic patterns to design very long masks that guarantee at least one hit [38]. As an alternative to hits, one may instead (or additionally) consider positions covered by hits [4, 30] and optimize the number of guaranteed covered positions.

Probabilistic setting and sensitivity of masks. In the probabilistic setting, one does not specify a fixed number of changes, but uses a random model of changes over the sequence, such a Bernoulli model: We compare two assumed homologous (or unrelated) sequences without indels that may differ at each position with some probability, independently of the other positions. A key computational task is to compute the probability distribution of the random number of hits [1] and specifically the probability of at least one hit (the *sensitivity* of the mask) vs. zero hits. These probabilities depend on the parameters of the random model. Methods integrating over different parameter values [7] and even symbolic approaches [31] have been developed. One may also design seeds to maximize sensitivity contrast between a homology model (small change probability per position, high sensitivity desired) and a background model (3/4 or similarly large change probability per position, low sensitivity desired) [15]. It has also been shown that sensitivity/selectivity trade-offs may be further improved by requiring a certain number of hits instead of a single hit [8]. Similarly to the decision problem in the lossless setting, computing the sensitivity of a mask is NP-hard [25], and also (exponential-time) dynamic programming algorithms over certain automata can be used to compute the sensitivity [19, 16, 27]. The search for high-sensitivity masks either exhaustively enumerates all masks (for fixed k, w), or employs heuristics [17, 18, 15], especially if multiple masks are considered at once. We note that many of the computational results in the past were limited to small parameter values, i.e. $k \leq 15, w \leq 22$, and short sequences of length $n \leq 64$.

Our contributions. Our approach is related to but different from the lossless setting and considers a *maximin optimization* approach; we are *not* using a probabilistic model of sequence similarity. We consider both the number of hits and hit-covered positions, as already the initial work in this field [4] did, but we are able to produce results on longer masks and sequences. We evaluate the quality of a single mask by its worst-case performance (number of hits or covered positions) by considering the two combinatorial minimization problems below, and then ask for the best mask(s) that maximize these minima.

MinHits For a given sequence length n (a typical short read, or part of a longer read) and a given number c of allowed sequence changes (substitutions), and a given spaced seed pattern (mask), find the worst-case distribution of the c changes that minimizes the number of seed *hits* inside the sequence (formal definitions below).

MinCov Under the same assumptions, find the distribution of the c changes that minimizes the number of sequence positions covered by the hits. (This is an alternative notion of “worst-case”.)

Solving the above two problems for a given mask yields its worst-case change distributions. This means that the resulting number of hits (resp. covered positions) are guaranteed for this mask for *every* distribution of c changes across any sequence of length n . Of course, if

the number of changes c is chosen too large, the objective value for both problems is zero, so there is a limited range of interest for parameter c . The more interesting problem, however, is to find a “best” mask where these minima are maximal among a specified class of masks.

MaxMinHits Given a sequence length n , a number of changes c , and a set \mathcal{M} of masks, find the masks that maximize the MinHits objective among all masks in \mathcal{M} .

MaxMinCov Under the same assumptions, find the masks that maximize the MinCov objective in \mathcal{M} .

We shall now define the necessary preliminaries and provide an example; then we give formal definitions of the above problems. We solve them for practically relevant seed parameters (k, w) , sequence lengths n and numbers of changes c . As a result, we obtain highly substitution-tolerant masks with guarantees that do not only hold with high probability but always. We illustrate the properties of these masks on a specifically designed simulated dataset of reads from the human genome.

2 Methods

We start by defining the necessary terms to work with spaced seeds; in particular, we define *masks* of a given *shape* (k, w) . Then, we introduce the optimization problems of computing the worst-case set of change positions, and the problem of finding the best mask among all masks of shape (k, w) . These problems are formulated as integer linear programs (ILPs).

2.1 Definitions

Basics. We write $[n]$ for the set of integers $\{0, \dots, n-1\}$. Indexing of strings starts at 0, so $s = (s_0, \dots, s_{n-1}) = (s_i)_{i \in [n]}$, where the length of s is $|s| = n$.

We consistently use a single (reference) sequence s in this manuscript; however, multiple sequences (e.g., chromosomes) are covered by concatenating them with separator characters $\$$, ignoring all (spaced) k -mers containing a separator, and translating between single sequence positions and chromosome-position-pairs accordingly.

If s and t are strings of equal length $|s| = |t| = n$, then their *Hamming distance* $d_H(s, t)$ is the number of indices $i \in [n]$ where $s_i \neq t_i$.

Masks of shape (k, w) . Given two integers $w \geq k \geq 2$, a (k, w) -mask is a string μ of length w over the alphabet $\{\#, _ \}$ that contains exactly k times the character $\#$ and $w - k$ times the character $_$. The positions marked $\#$ are called *significant*, and the positions marked $_$ are called *insignificant*, jokers, “don’t care” positions, or spaces. We call k the *weight* of the mask and w its *width* or *window length*. The pair (k, w) is called the *shape* of the mask. A mask μ may also be represented as the tuple κ of significant positions: $\kappa = \{j : 0 \leq j < w \text{ and } \mu_j = \#\}$.

We require that $\mu_0 = \mu_{w-1} = \#$, because otherwise the mask could be shortened by removing the insignificant characters at each end to obtain an equivalent smaller mask (disregarding matches at sequence boundaries). With this constraint, there are $\binom{w-2}{k-2}$ different masks of shape (k, w) .

A mask μ is *symmetric* if $\mu = \text{rev}(\mu) := \mu_{n-1} \dots \mu_1 \mu_0$. We consider only symmetric masks in this work because then we can work with canonical spaced k -mers, as the mask is the same on both DNA strands. We also restrict ourselves to odd k to avoid any issues with k -mers that are equal to their reverse complement. This, together with symmetry, implies that w must also be odd. For symmetric masks with odd k and w , we must have $\mu_m = \#$

Three examples with 3 changes in 27 positions:			
X.....X.....X.....X.....X.....X.....X.....XX.....
examples	#####	##_###_##	##_#_#_##
of	#####	##_###_##	##_#_#_## (hit1)
shifted	#####	##_###_##	##_#_#_## (hit2)
masks	#####	##_###_##	##_#_#_##
	(all shifts overlap an X)	(all shifts overlap an X)	(2 hits exist)

■ **Figure 1** Examples for weight $k = 7$, $c = 3$ changes and sequence length $n = 27$. For contiguous 7-mers ($k = w = 7$, mask #####, left), we can place $C^\dagger = 3$ changes at (0-based) positions $X = \{6, 13, 20\}$ to change all 7-mers. For mask ##_###_## (middle) with $(k, w) = (7, 11)$, we have the same result with the same placement of changes. However, using the $(7, 11)$ -mask ##_#_#_## (right), we tolerate $C^* = 3$ changes and always guarantee at least 2 hits (unchanged spaced k -mers) and at least 10 covered positions. The rightmost example shows one worst-case positioning of the 3 changes to minimize hits; it is not trivial to see this by eye, but is a result of the **MinHits** optimization problem in Sec. 2.2.

for the middle position $m = (w - 1)/2$, and there are $\binom{(w-3)/2}{(k-3)/2}$ such masks. They can be enumerated by enumerating all possible left halves and mirroring them to obtain the right halves.

We write $\mathcal{M}(k, w)$ for the set of all mask tuples κ (with significant ends) of shape (k, w) and $\mathcal{S}(k, w)$ for the subset of all symmetric ones. For example, $\kappa = (0, 2, 5, 8, 10) \in \mathcal{S}(5, 11)$, corresponding to $\mu = \#_#_#_#_#$. For $k = 25$ and $w = 37$, we find $|\mathcal{M}(k, w)| = 834\,451\,800$ and $|\mathcal{S}(k, w)| = 12\,376$. It becomes clear that symmetry is a strong constraint.

κ -mers of a sequence. Given a mask μ of shape (k, w) with its corresponding offset tuple κ and a string s of length n over an alphabet (e.g., the DNA or protein alphabet), we obtain $n - w + 1$ *spaced k -mers* from s : The i -th spaced k -mer g_i , for $i \in [n - w]$, is obtained by concatenating the significant positions when the mask is applied at positions $i, \dots, i + w - 1$ of s , i.e., $g_i = (s_{i+j})_{j \in \kappa}$. We also say that each such string g_i is a μ -mer or κ -mer of s , depending on whether we are referring to the mask's string representation μ or tuple representation κ . In this sense, a (contiguous) k -mer of s is a $\kappa = (0, 1, \dots, k - 1)$ -mer of s .

Effects of sequence changes on κ -mers. When we change a single character in a sequence s , then up to k of the κ -mers may be changed (less if the changed position is near either end of the sequence). We say that the κ -mer starting at position $p \in [n - w + 1]$ (κ -mer number p for short) is *changed* or *destroyed* if any of the positions $\{p + j : j \in \kappa\}$ is changed. Otherwise, the κ -mer is *unchanged* or *unaffected*. An unaffected κ -mer matches or *hits* the sequence.

We say that sequence position $i \in [n]$ is *covered* (by an unaffected κ -mer) if there exists an unaffected κ -mer starting at some position $p \leq i$ such that $i = p + j$ for some $j \in \kappa$. In other words, if the κ -mer starting at position p is unaffected, then all positions $p + j$ for $j \in \kappa$ are covered.

2.2 Optimization Problems

We introduce the different optimization problems that we consider in this work. For a given mask κ of shape (k, w) and given sequence length n , it asks how many changes $C^* = C^*(\kappa, n)$ can be made at most in order to guarantee at least one unchanged κ -mer (for all possible placements of c changes among n positions). Equivalently, we may find the smallest C^\dagger such

that C^\dagger changes, placed strategically, change all κ -mers; then $C^* = C^\dagger - 1$. Masks of the same shape (k, w) may have different C^* -values (Figure 1), but even masks with the same (highest) C^* -value can be further distinguished.

This leads to another optimization problem for a given mask κ , sequence length n , and number of allowed changes c (where $c \leq C^*(\kappa, n)$): Find a worst-case positioning of the changes among the n sequence positions. There are two variants “MinHits” and “MinCov” of the problem that respectively ask to place the changes to minimize

1. (**MinHits**) the number of (starting positions of) unaffected κ -mers (“hits”);
2. (**MinCov**) the number of covered positions.

The corresponding decision problems, “does there exist a placement of c changes such that there are zero hits (equivalent to zero covered positions)?”, has been proven to be NP-complete [29], but exact solutions can be obtained in practice by ILP solvers [14], as we demonstrate below.

Given a shape (k, w) , i.e., weight k and window length w , we seek a (symmetric) mask κ of the given shape that *maximizes*, among all such (symmetric) masks, the minimized worst-case number of hits (**MaxMinHits**) or covered positions (**MaxMinCov**). These are *maximin* problems, and the approach that we take to solve them is to solve MinHits and MinCov exhaustively for all masks of the given shape, and then pick the best mask.

2.3 Integer Linear Program Formulations

We formulate the optimization problems MinHits and MinCov as integer linear programs (ILPs). The used integer variables are binary, i.e., take only values in $\{0, 1\}$. We introduce the complete set of parameters and variables here, even though some of the problems use only a subset. The given (constant) parameters are:

- n : length of the sequence (typically $n = 100$, corresponding to the length of a short read);
- c : number of allowed changes;
- k : weight of the mask, number of significant positions, typically $k \geq 21$ on a mammalian genome;
- w : window length, $k \leq w \leq n$
- κ : k -tuple of offsets with $0 \in \kappa$, $w - 1 \in \kappa$, satisfying the symmetry condition $i \in \kappa \iff w - 1 - i \in \kappa$;

We use the following *binary* variables:

- $x = (x_i)_{i \in [n]}$, indicators of changed positions: $x_i = 1$ if and only if sequence position i is changed;
- $y = (y_p)_{p \in [n-w+1]}$, indicators of (starting positions of) unchanged κ -mers: $y_p = 1$ if and only if the κ -mer starting at position p is unchanged (i.e., none of its significant positions is changed),
- $z = (z_i)_{i \in [n]}$, indicators of covered positions: $z_i = 1$ if and only if sequence position i is covered by an unchanged κ -mer.

As seen here, we use $i \in [n]$ for indexing sequence positions, $p \in [n - w + 1]$ for indexing starting positions of κ -mers, and j for an element of κ .

2.3.1 TolChg: Tolerated number of changes for a mask

For a given n and κ , we minimize the required number of changed positions $C^\dagger = C^\dagger(\kappa, n)$ to change *all* κ -mers, i.e., at least one significant position of each κ -mer coincides with a changed position. The number of tolerated changes is then $C^*(\kappa, n) = C^\dagger(\kappa, n) - 1$. We only need the binary x -variables (see above). The following ILP computes C^\dagger .

$$\text{Minimize } \sum_{i \in [n]} x_i, \quad (1)$$

$$\text{such that } \sum_{j \in \kappa} x_{p+j} \geq 1 \quad \text{for all } p \in [n - w + 1]. \quad (2)$$

2.3.2 MinHits

Assuming that a number of changes $c \leq C^*(\kappa, n)$ is specified, we compute the worst-case distribution of these c changes in a sequence of length n that minimizes the number of unchanged κ -mers. We need the binary x - and y -variables.

$$\text{Minimize } \sum_{p \in [n-w+1]} y_p, \quad (3)$$

$$\text{such that } \sum_{i \in [n]} x_i = c, \quad (4)$$

$$y_p \geq 1 - \sum_{j \in \kappa} x_{p+j} \quad \text{for all } p \in [n - w + 1], \quad (5)$$

$$y_p \leq 1 - x_{p+j} \quad \text{for all } p \in [n - w + 1], j \in \kappa. \quad (6)$$

Here, Eq. (4) ensures that we use exactly c changes, the group of inequalities (5) ensures that $y_p = 1$ if all positions of the κ -mer starting at p are unchanged, and (6) conversely ensures that $y_p = 0$ if any position of that κ -mer is changed. This last group of inequalities (6) is unnecessary because we minimize the sum of y_p , driving $y_p = 0$ automatically whenever possible.

2.3.3 MinCov

Again, for given $c \leq C^*(\kappa, n)$, we minimize the number of positions covered by unchanged k -mers. A position is covered if it is contained in at least one unchanged k -mer. This ILP is an extension of the MinHits formulation, additionally relating the z - to the y -variables, with a different objective function. We thus need the x -, y - and z -variables.

$$\text{Minimize } \sum_{i \in [n]} z_i, \quad (7)$$

such that (4), (5), (6) hold, and additionally

$$z_{p+j} \geq y_p \quad \text{for all } p \in [n - w + 1], j \in \kappa, \quad (8)$$

$$z_i \leq \sum_{j \in \kappa, i-j \geq 0} y_{i-j} \quad \text{for all } i \in [n]. \quad (9)$$

Inequalities (8) force the z -variables of all significant positions covered by an unaffected κ -mer to 1, whereas (9) forces the z -variables of positions not covered by any such κ -mer to zero. This last group of inequalities (9) is unnecessary because of the direction of the objective function.

2.3.4 ILP Sizes

The ILPs presented here are of moderate size for typical values of n , c , k and w (Table 1). Modern commercial solvers typically solve an instance for a given mask κ within a few seconds. The challenge, however, is the large number of (even symmetric) masks of shape (k, w) , as the ILPs have to be solved for each of the masks. Solution times and optimal masks are given in the Results section.

■ **Table 1** Considered ILPs and their sizes (vars.: used variables, see text; obj.: objective function; #: number of). The number of constraints counts only necessary constraints, without (6) for MinHits and without (9) for MinCov.

ILP	vars.	obj.	constraints	#variables	#constraints
TolChg	x	(1)	(2)	n	$n - w + 1$
MinHits	x, y	(3)	(4), (5)	$2n - w + 1$	$n - w + 2$
MinCov	x, y, z	(7)	(4), (5), (6), (8)	$3n - w + 1$	$(2k + 1)(n - w + 1) + 1$

3 Results

We first discuss several interesting masks for typical short reads of length $n = 100$ and their properties in Section 3.1. These results are also useful for longer reads, as then they hold for *every* substring of length 100. The Supplement contains tables with optimal masks for many shapes (k, w) for $n = 100$ and various number of changes $c \in \{3, 4, 5, 6, 7\}$.

In Section 3.2, we provide information on the solving times of the ILPs from Section 2.3.

Section 3.3 describes how we designed sets of reads that are on the one hand easily mappable to the telomere-to-telomere (t2t) reference genome [32], as the regions they map to are non-repetitive, but on the other hand are hard to map because they contain 5 or 6 substitutions spread over the entire read. In Section 3.4, we evaluate how successfully established read mappers and a simple alignment-free approach based on our masks can map these reads back to the reference. These results should be understood as a proof-of-concept for the usefulness of optimized masks, and not as claims that we have a better read mapper than existing ones.

3.1 Examples of Optimal Masks

We point out a few examples of properties of contiguous vs. designed spaced masks with the same value of k .

For $n = 100$ and (standard) 21-mers, placing 4 changes at (0-based) positions (20, 41, 62, 83) affects all 21-mers, so the standard k -mer mask with $k = w = 21$ (mask D in Table 2) only tolerates $C^* = 3$ changes. In contrast, the $(k, w) = (21, 25)$ -shaped masks E and F in Table 2, #####_####_###_#####_##### and #####_#####_###_#####_####, both tolerate 4 changes and then guarantee at least 8 hits and a minimum of 44 covered positions out of 100, for *every* distribution of 4 changes. It is noteworthy that the change distribution(s) that achieve(s) the worst case of 8 hits may be very different from the distribution(s) that achieve(s) 44 covered positions. In fact, one can show (with modified ILPs; not explained here) that both of these masks, when we have only 8 hits, we have at least 60 covered positions, and when we have only 44 covered positions, we have at least 20 hits. In any case, *every* distribution of 4 changes gives at least 8 hits *and* 44 covered positions.

Better still, both masks E and F tolerate even 5 changes, with at least 3 hits and 33 covered positions. There is no mask with $k = 21$ and $21 \leq w \leq 41$ that guarantees more than 8 hits for 4 changes, but the (21, 33)-shaped mask G, ###_##_#_#_#_###_#_###_#_#_#_###, guarantees at least 55 covered positions but only 4 hits.

These examples illustrate the benefit of spaced seeds. Several other interesting masks are collected in Table 2, which are also used for our computational experiments. Full tables of optimal masks for different shapes (k, w) for $c \in \{3, 4, 5, 6, 7\}$ changes on $n = 100$ positions are included in the Supplement.

■ **Table 2** Contiguous k -mers vs. well performing masks. Each row shows a mask and its properties for $c \in \{3, 4, 5\}$ changes in $n = 100$ positions. The single-letter label A–Q is used to identify the mask in other tables and figures. The masks D–G for $k = 21$ are discussed in Sec. 3.1. The shape (k, w) is given in addition to the mask for convenience; the remaining columns show MinHits (MH) and MinCov (MC) optimal objective values for different values of c . Values of zero are highlighted as **0**; maximal known values for the given k (for any w) are highlighted in **bold green**.

	(k, w) shape	mask	$c = 3$		$c = 4$		$c = 5$	
			MH	MC	MH	MC	MH	MC
A	(19, 19)	#####	25	43	6	24	0	0
B	(19, 23)	####_####_###_####_####	21	68	11	48	6	42
C	(19, 23)	####_###_#####_###_####	21	68	11	48	6	42
D	(21, 21)	#####	17	37	0	0	0	0
E	(21, 25)	#####_####_###_####_#####	13	65	8	44	3	33
F	(21, 25)	#####_#####_###_#####_####	13	65	8	44	3	33
G	(21, 33)	###_##_#_#_#_###_#_###_#_#_#_###_###	15	63	4	55	2	31
H	(23, 23)	#####	9	31	0	0	0	0
I	(23, 35)	###_####_#_###_#_###_#_####_###	12	59	5	48	2	34
J	(23, 37)	#####_#_#_#_#_#####_#_#_#_#_#####	11	55	4	45	2	34
K	(23, 37)	###_###_#_#_###_#_#_#_###_#_###_###	12	58	6	47	2	34
L	(25, 25)	#####	1	25	0	0	0	0
M	(25, 35)	#####_###_#_###_#_###_#####	9	55	4	42	0	0
N	(25, 37)	#####_#_#_###_###_###_###_#_#_#####	9	53	4	45	0	0
O	(25, 37)	#####_#_#_####_###_#####_#_#_#####	8	54	2	47	0	0
P	(27, 27)	#####	0	0	0	0	0	0
Q	(27, 39)	##_####_###_#_###_#_###_#_####_####_##	7	52	2	42	0	0

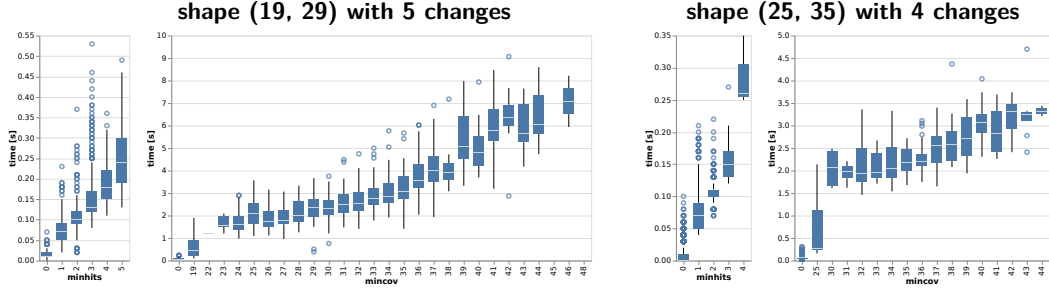
3.2 ILP Solving Times

We used Gurobi 10.0.3 [14] to solve the ILPs from Table 1 on a AMD Ryzen 9 5950X 16-core Processor with hyperthreading (32 logical threads) and 128 GB RAM (which is not needed). Figure 2 shows an overview of solving times for all 1287 symmetric masks of shape (19, 29) for $c = 5$ changes and sequence length $n = 100$, and all 4368 symmetric masks of shape (25, 35) for $c = 4$ changes (none of these masks tolerates 5 changes) and $n = 100$. There are a few observations to be made.

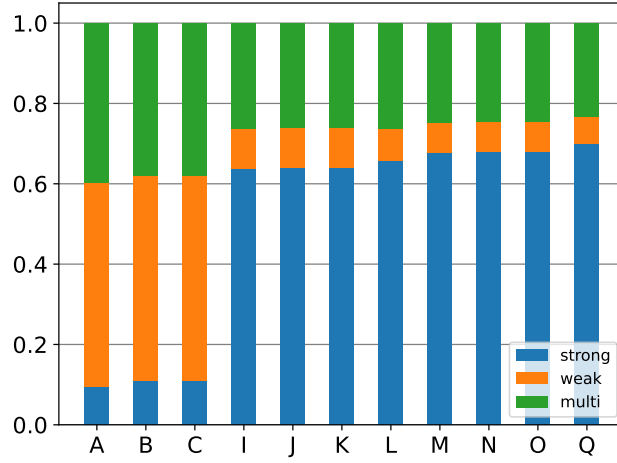
First, solving times are very fast (under half a second) for the MinHits problem and reasonable (under 10 seconds) for the MinCov problem for the chosen parameters. Indeed, the MinHits problem is considerably smaller (Table 1), easier and faster to solve than the MinCov problem.

Second, the solving time depends on the optimal objective value. An objective value of zero (there exists a change distribution for which all κ -mers are changed) is found quickly. As a tendency, the better the mask (i.e., the higher the MinHits or MinCov objective value), the harder it is to find the corresponding worst-case change distribution or prove its optimality.

Third, unfortunately, increasing c or n can have a drastic effect on solving times. This should not be surprising: We are looking for the worst-case placement of c changes among n positions, for which there are $\binom{n}{c}$ possibilities, where $\binom{100}{4} \approx 4 \cdot 10^6$ and $\binom{100}{5} \approx 75 \cdot 10^6$. While the time increase between 4 and 5 changes seems moderate in Fig. 2, solving for much larger values of n and c than provided in the Supplement is currently not feasible in a comprehensive manner. For example, solving the ILPs for the best (19, 29)-mask



■ **Figure 2** Solving times (y-axis) of the MinCov and MinHits ILP with $n = 100$ for masks of shape (19, 29) with 5 changes (two left plots) and for masks for shape (25, 35) with 4 changes (two right plots). The x-axes show the distinct objective values of the MinHits and MinCov problems. Empty boxplots (e.g. shape (19, 29) with highest MinCov objective value 48) occur if too few masks have this value to produce a meaningful boxplot.



■ **Figure 3** Fractions of positions in the t2t reference genome where strongly unique (blue), weakly unique (orange) and non-unique (green, “multi”) κ -mers start for selected masks from Table 2.

###_###_#_#_###_#_#_###_### for $n = 100$ and $c = 5$ (guaranteeing 5 hits or 48 covered positions) takes 0.28 seconds for MinHits, and 9.03 seconds for MinCov. Doubling these to $n = 200$ and $c = 10$ increases the time to 41 seconds for MinHits (a factor of 146) and to 5200 seconds for MinCov (a factor of 576).

However, we may argue that using $n = 100$ (corresponding to a typical short read length) is sufficient because then the guarantees hold for *every* substring of length 100 of longer reads, and by the pigeonhole principle, if 50 changes are distributed over 1000 positions, there must exist at least one length-100 substring with at most 5 changes. In general, ILP solving times may depend on the decisions taken by the solver and may vary even for the same problem (if randomization is used) and across problems of comparable size and difficulty, and also change from version to version of the solver.

3.3 Creating Challenging Read Datasets

Clearly, established read mappers, such as bwa-mem2 [39], minimap2 [22, 23], or more recent developments like strobealign [37] work well in most settings; otherwise, they would not be used. We acknowledge that the optimal masks presented in Table 2 and their thresholds are

mostly a theoretical result and perhaps of limited practical value for production pipelines in bioinformatics core facilities. Nonetheless, to investigate the case of highly diverse sequence regions, we decided to create datasets that pose a challenge to seed-based mappers (including our designed masks) because no long exact seeds exist.

We created two datasets, each of 5 million short reads of length $n = 100$, from the autosomes of the t2t human reference genome [32]. The intervals of origin were chosen such that the reads, when unchanged, are easy to map back to the genome, in the sense that repetitive regions were excluded. Precisely, we ensured that all canonical k -mers of each selected read for $k = 27$, are unique in the reference. In fact, we picked only regions where each 27-mer is not only unique, but also does not have a Hamming distance 1 neighbor elsewhere in the reference. Such k -mers have been called *strongly unique* [42]. Unique k -mers that are not strongly unique are weakly unique. As a rule of thumb, for $k \geq 23$, a large fraction of the k -mers in the t2t reference genome is strongly unique, and for $k \geq 27$, this fraction saturates [42], which motivated our choice of $k = 27$.

Figure 3 contains statistics about how many positions in the t2t reference genome we find a strongly unique, weakly unique or non-unique κ -mer for selected masks from Table 2. Clearly, $k = 19$ (A, B, C) does not yield many strongly unique κ -mers, but still an overall high fraction (over 60%) of unique κ -mers. For $k = 23$ (I, J, K), we already have strongly unique κ -mers at over 60% of the genome positions, and a few more positions with weakly unique κ -mers. Increasing k to 25 (L, M, N, O) or to 27 (Q) only slightly increases the fraction of strongly unique κ -mers further; the fraction of non-unique (“multi”) κ -mers stays more or less constant. We see that selecting reads from regions with only strongly unique 27-mers is not overly restrictive.

In the unchanged reads, any single contiguous 27-mer would be sufficient to reliably place the read at the correct location in the reference genome. While the unmodified reads would thus be trivial to map, we introduce a large number of changes ($c = 5$ or $c = 6$) to obtain the actual datasets.

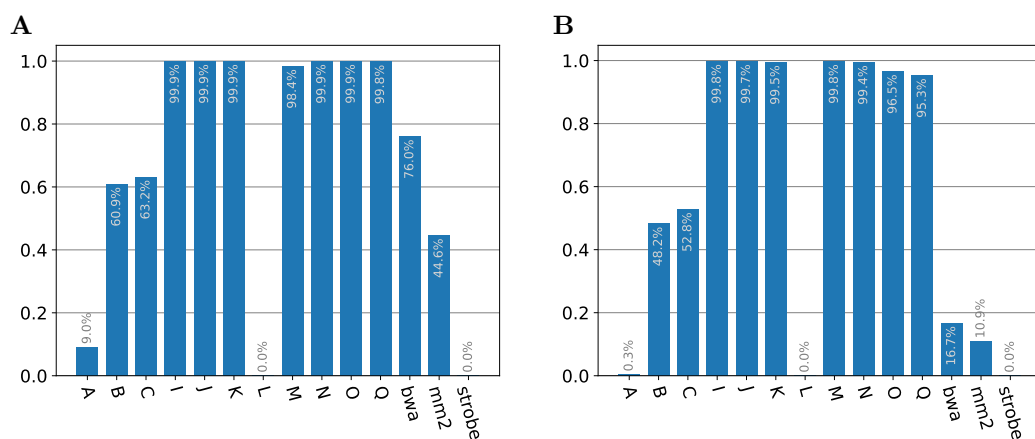
In the first dataset, we introduced $c = 5$ changes ($A \leftrightarrow T$, $C \leftrightarrow G$) into each read at approximately equidistant positions. More precisely, for each read, we modified each of the five 0-based positions (16, 33, 50, 67, 84) by a random offset uniformly chosen from $\{-3, \dots, 0, \dots, 3\}$, and then complemented the base at the 5 obtained positions. In the second dataset, we proceeded similarly, but with $c = 6$ changes at randomly modified positions obtained from (14, 28, 43, 57, 72, 86) with uniformly chosen random offsets between -3 and $+3$.

Clearly, this mapping task is challenging for all of the approaches: Using 5 or 6 changes makes it hard for the existing mappers to find a good seed, and none of our masks offers any guarantees for 6 changes either: All MinHits and MinCov objective values are zero for $c = 6$. Note that some of our masks do offer guarantees for $c = 5$.

3.4 Evaluation

To attempt to map the modified reads back to their interval of origin in the reference genome, we used bwa-mem2 v2.2.1 [39], minimap2 v2.26 [22, 23] and strobealign v0.11.0 [37] with their default parameters. For these tools, a read was considered uniquely mapped if the resulting BAM file contained a unique alignment; it was considered correctly mapped if that unique alignment started at the correct position (with a tolerance of half the read length).

For our selection of masks from Table 2, we did not create a read mapper for spaced k -mers within the scope of this work, but used a simple alignment-free approach. We created a positional index, implemented as a multi-way bucketed Cuckoo hash table [40, 41] for each



■ **Figure 4** Fraction of correctly mapped reads from the challenge datasets with 5 or 6 introduced changes (**A**: 5 changes; **B**: 6 changes; cf. Sec. 3.3). Mask labels refer to Table 2. Tool labels bwa, mm2 and strobe refer to the established tools bwa-mem2, minimap2 and strobealign.

mask, that for each unique canonical κ -mer stores its unique location (chromosome and position) and whether it is strongly (vs. weakly) unique. For non-unique κ -mers, it stores the special value **non-unique**.

To attempt to “map” a read with a mask, we query the genome position of each κ -mer in the read and, if unique, subtracted the relative position within the read, to obtain a virtual start position of the read in the genome. We count (for at most 20 different candidate starting positions) how many strongly and weakly unique κ -mers indicate each candidate position, and aggregate counts to the dominant position (highest count) inside an interval of half the read length. The read is considered uniquely mapped if we find at least two strongly unique or four unique κ -mers at the dominant position, and that position is the only remaining one with strong κ -mers after aggregation. The read is correctly mapped if the unique position is the correct one in the genome (with a tolerance of half the read length).

Figure 4 shows the fraction of correctly mapped reads for selected masks and each tool for both datasets. The alignment tools bwa-mem2 and minimap2 struggle particularly with the dataset using 6 changes, because they often do not find a sufficiently long seed to initiate alignment. For the dataset with 5 changes, bwa-mem2 still performs relatively well. Both minimap and strobealign suffer more because they do not examine all k -mers seeds but use a minimizer-based sampling approach.

Masks representing contiguous k -mers (A: 19-mers, L: 25-mers) also perform extremely badly because almost no unchanged 19-mer or 25-mer exists in these reads. While some 19-mers remain unchanged when only 5 changes are introduced (depending on the chosen random offsets), 19-mers are not specific enough to uniquely place a read within the genome using an alignment-free approach. Note that bwa2-mem also requires an exact match with a minimum length of 19 to initiate further alignment. Since bwa-mem2 also considers non-unique 19-mers and investigates up to 1000 positions, it is more successful than our simplistic approach that only uses unique 19-mers. The change-tolerant spaced 19-mers (B, C) perform much better than the contiguous 19-mer (A), but still not very well. The limiting factor here is the unavailability of sufficiently many (strongly) unique spaced 19-mers (Fig. 3).

Our masks with $k = 23$ (I, J, K) perform almost equally well for both 5 and 6 changes, even though they guarantee only 2 hits for 5 changes and provide no hit guarantee for 6 changes. Because the change distribution is semi-random, we typically do not see the masks’ worst-case distributions and therefore usually obtain the required strongly unique k -mer hits even for 6 changes.

The spaced 25-mer masks (M, N, O) and the spaced 27-mer mask (Q) do not give any guarantees for 5 or 6 changes. Nonetheless, a large number of reads is placed correctly in both cases, again because the semi-random placement of changes does with high probability not correspond to the worst-case distribution.

In summary, we find that selected spaced masks with weight $k = 23$ or $k = 25$ are well suited to map reads with high substitution rates (5%, 6%) back to the correct location in genome, even in an alignment-free manner, whereas this task does not work well with contiguous k -mers (even for smaller k), which are used as seeds for most established tools. The low mapping rates of strobealign are surprising, but we note that it was designed mostly with indel tolerance in mind. Of course, the practical significance of this evaluation is limited: The datasets were designed to point out the difficulties of classical tools with certain distributions of large numbers of substitutions. Nonetheless, it is noteworthy that by using certain spaced masks as seeds instead of contiguous k -mers, one can achieve a high built-in substitution tolerance in alignment-free methods.

3.5 Software and Data

With this article, we provide the following tools and data, further explained in the repository's README file¹:

1. a Python program that calls the Gurobi optimizer [14], for which a (free academic) license is required for computing the worst-case change distributions of a mask or a set of masks of the same shape,
2. a just-in-time compiled Python program to index the κ -mers of a genome in FASTA format (`gapmap index`), which is a modification of our spaced k -mer counter `hackgap` [41],
3. a just-in-time compiled Python program to map a FASTQ file of reads using a pre-computed κ -mer index (`gapmap map`).

We furthermore provide the following data:

4. a list of worst-case optimal masks for different shapes (k, w) and number of changes c for $n = 100$ in the Supplement,
5. our modified t2t reference genome ("analysis set") consisting of the t2t genome [32], an Epstein-Barr virus (EBV) sequence and a PhiX sequence,²
6. the challenge datasets of Sec. 3.3 of 100-bp reads with 5 changes³ and 6 changes⁴. The FASTQ headers contain the true origins and the positions of the introduced changes.

4 Discussion and Conclusion

With this work, we open a new chapter in spaced seed design. Instead of considering the probability or guarantee of *at least one hit*, we use a combinatorial approach and explicitly compute the worst-case distribution of c changes (substitutions) among n positions to *minimize the number of hits or the number of positions covered by hits*. The integer linear program formulations we use for this task are reasonably efficient for $n = 100$ and allow exhaustive enumeration of all symmetric masks for a wide range of shapes (k, w) ; but they show their limits for longer sequences: Already for $n = 200$, the solving times for a single

¹ <https://gitlab.com/rahmannlab/seed-optimization>

² <https://kingsx.cs.uni-saarland.de/index.php/s/CcggyrNZWpFoZWj>

³ <https://kingsx.cs.uni-saarland.de/index.php/s/nHJbaAxi3ZGApEG>

⁴ <https://kingsx.cs.uni-saarland.de/index.php/s/mA9cXMYcgHsZXFb>

mask prohibit evaluating most shapes of interest. Further engineering, such as the inclusion of starting and improvement heuristics, and exploitation of the regular constraint structure, may lead to substantial performance improvements.

We focus on shapes (k, w) for which most κ -mers in a mammalian genome can be expected to be unique or even strongly unique with the idea that a few strongly unique κ -mer matches suffice to map a read to a specific position of the reference genome. As demonstrated, this works in the human genome for $k \geq 23$, for which many strongly unique k -mers exist. This is different from previous work that focused on small k , and used matching κ -mers as seeds to initiate alignments at potentially many different positions where the κ -mer occurs. The approach based on strongly unique κ -mers can be orders of magnitude faster and at the same time highly reliable, as shown by our challenge dataset. Thus, strongly unique worst-case-optimal spaced seeds provide an orthogonal method against reference bias besides graph genomes.

We found that good worst-case masks are rare, so we cannot expect to find a good mask randomly. Also, which mask is optimal may differ for different values of c (for the same k , w and n). This suggests using a combination of several masks, which would also increase the chance of seeing more strongly unique κ -mers (across all used masks). The problem then becomes finding an optimal combination of masks (perhaps even of different shapes). This has already been considered within the probabilistic approach, but an exhaustive enumeration of all combinations is not feasible, so heuristics have been proposed, which may be adapted to worst-case mask design in the future.

A drawback of spaced seeds is their rigidity over their width w , i.e., they tolerate substitutions, but not insertion- or deletion-type differences. There is a trade-off: A longer width may lead to more hits or higher coverage, but also offers less tolerance against insertions or deletions. While strobemers were developed with indel tolerance in mind, they did not perform well in the adversarial setting considered here. From our point of view, it remains a challenging open problem to define and design alternative general error-tolerant types of seeds that provide worst-case guarantees instead of probabilistic guarantees.

There are other open questions as well: We are currently unable to predict the performance of a mask (e.g., from substrings, motifs, etc.). It would be interesting to use machine learning approaches to predict which masks have the potential to yield high MinHits or MinCov values to avoid running all ILPs.

References

- 1 Gary Benson and Denise Y. F. Mak. Exact distribution of a spaced seed statistic for DNA homology detection. In Amihoud Amir, Andrew Turpin, and Alistair Moffat, editors, *String Processing and Information Retrieval, 15th International Symposium, SPIRE 2008, Melbourne, Australia, November 10-12, 2008. Proceedings*, volume 5280 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2008. doi:10.1007/978-3-540-89097-3_27.
- 2 B. Brejová, D. G. Brown, and T. Vinar. Optimal spaced seeds for homologous coding regions. *J Bioinform Comput Biol*, 1(4):595–610, January 2004. doi:10.1142/S0219720004000326.
- 3 Karel Brinda. Languages of lossless seeds. In Zoltán Ésik and Zoltán Fülöp, editors, *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014*, volume 151 of *EPTCS*, pages 139–150, 2014. doi:10.4204/EPTCS.151.9.
- 4 Stefan Burkhardt and Juha Kärkkäinen. Better filtering with gapped q-grams. *Fundam. Informaticae*, 56(1-2):51–70, 2003. URL: <http://content.iospress.com/articles/fundamenta-informaticae/fi56-1-2-04>.

- 5 K. Brinda, M. Sykulski, and G. Kucherov. Spaced seeds improve k-mer-based metagenomic classification. *Bioinformatics*, 31(22):3584–3592, November 2015. doi:10.1093/BIOINFORMATICS/BTV419.
- 6 N. C. Chen, B. Solomon, T. Mun, S. Iyer, and B. Langmead. Reference flow: reducing reference bias using multiple population genomes. *Genome Biol*, 22(1):8, January 2021.
- 7 W. H. Chung and S. B. Park. Hit integration for identifying optimal spaced seeds. *BMC Bioinformatics*, 11 Suppl 1(Suppl 1):S37, January 2010.
- 8 L. Egidi and G. Manzini. Multiple seeds sensitivity using a single seed with threshold. *J Bioinform Comput Biol*, 13(4):1550011, August 2015.
- 9 B. Ekim, K. Sahlin, P. Medvedev, B. Berger, and R. Chikhi. Efficient mapping of accurate long reads in minimizer space with mapquik. *Genome Res*, 33(7):1188–1197, July 2023.
- 10 Martin Farach-Colton, Gad M. Landau, Süleyman Cenk Sahinalp, and Dekel Tsur. Optimal spaced seeds for faster approximate string matching. *J. Comput. Syst. Sci.*, 73(7):1035–1044, 2007. doi:10.1016/J.JCSS.2007.03.007.
- 11 C. Firtina, J. Park, M. Alser, J. S. Kim, D. S. Cali, T. Shahroodi, N. M. Ghiasi, G. Singh, K. Kanellopoulos, C. Alkan, and O. Mutlu. Blend: a fast, memory-efficient and accurate mechanism to find fuzzy seed matches in genome analysis. *NAR Genom Bioinform*, 5(1):lqad004, March 2023.
- 12 M. C. Frith and L. Noé. Improved search heuristics find 20,000 new alignments between human and mouse genomes. *Nucleic Acids Res*, 42(7):e59, April 2014.
- 13 E. Garrison and A. Guarracino. Unbiased pangenome graphs. *Bioinformatics*, 39(1), January 2023. doi:10.1093/BIOINFORMATICS/BTAC743.
- 14 Gurobi Optimization, LLC. Gurobi Optimizer reference manual, 2023. URL: <https://www.gurobi.com>.
- 15 L. Hahn, C. A. Leimeister, R. Ounit, S. Lonardi, and B. Morgenstern. rasbhari: Optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. *PLoS Comput Biol*, 12(10):e1005107, October 2016. doi:10.1371/JOURNAL.PCBI.1005107.
- 16 Inke Herms and Sven Rahmann. Computing alignment seed sensitivity with probabilistic arithmetic automata. In Keith A. Crandall and Jens Lagergren, editors, *Algorithms in Bioinformatics, 8th International Workshop, WABI 2008, Karlsruhe, Germany, September 15-19, 2008. Proceedings*, volume 5251 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 2008. doi:10.1007/978-3-540-87361-7_27.
- 17 L. Ilie, S. Ilie, and A. M. Bigvand. SpEED: fast computation of sensitive spaced seeds. *Bioinformatics*, 27(17):2433–2434, September 2011. doi:10.1093/BIOINFORMATICS/BTR368.
- 18 S. Ilie. Efficient computation of spaced seeds. *BMC Res Notes*, 5:123, February 2012.
- 19 Uri Keich, Ming Li, Bin Ma, and John Tromp. On spaced seeds for similarity search. *Discret. Appl. Math.*, 138(3):253–263, 2004. doi:10.1016/S0166-218X(03)00382-2.
- 20 Gregory Kucherov, Laurent Noé, and Mikhail A. Roytberg. Multiseed lossless filtration. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 2(1):51–61, 2005. doi:10.1109/TCBB.2005.12.
- 21 B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Methods*, 9(4):357–359, March 2012.
- 22 H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, September 2018. doi:10.1093/BIOINFORMATICS/BTY191.
- 23 H. Li. New strategies to improve minimap2 alignment accuracy. *Bioinformatics*, 37(23):4572–4574, December 2021. doi:10.1093/BIOINFORMATICS/BTAB705.
- 24 H. Li and R. Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26(5):589–595, March 2010. doi:10.1093/BIOINFORMATICS/BTP698.
- 25 Ming Li, Bin Ma, and Louxin Zhang. Superiority and complexity of the spaced seeds. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 444–453. ACM Press, 2006. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109607>.

- 26 M. J. Lin, S. Iyer, N. C. Chen, and B. Langmead. Measuring, visualizing and diagnosing reference bias with biastools. *bioRxiv*, September 2023. doi:10.1101/2023.09.13.557552.
- 27 Tobias Marschall, Inke Herms, Hans-Michael Kaltenbach, and Sven Rahmann. Probabilistic arithmetic automata and their applications. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 9(6):1737–1750, 2012. doi:10.1109/TCBB.2012.109.
- 28 R. Martiniano, E. Garrison, E. R. Jones, A. Manica, and R. Durbin. Removing reference bias and improving indel calling in ancient DNA data analysis by mapping to a sequence variation graph. *Genome Biol*, 21(1):250, September 2020.
- 29 François Nicolas and Eric Rivals. Hardness of optimal spaced seed design. *J. Comput. Syst. Sci.*, 74(5):831–849, 2008. doi:10.1016/J.JCSS.2007.10.001.
- 30 Laurent Noé and Donald E. K. Martin. A coverage criterion for spaced seeds and its applications to support vector machine string kernels and k -mer distances. *J. Comput. Biol.*, 21(12):947–963, 2014. doi:10.1089/CMB.2014.0173.
- 31 L. Noé. Best hits of 11110110111: model-free selection and parameter-free sensitivity calculation of spaced seeds. *Algorithms Mol Biol*, 12:1, 2017.
- 32 S. Nurk, S. Koren, A. Rhie, M. Rautiainen, A. V. Bzikadze, A. Mikheenko, M. R. Vollger, N. Altomose, L. Uralsky, A. Gershman, S. Aganezov, S. J. Hoyt, M. Diekhans, G. A. Logsdon, M. Alonge, S. E. Antonarakis, M. Borchers, G. G. Bouffard, S. Y. Brooks, G. V. Caldas, N. C. Chen, H. Cheng, C. S. Chin, W. Chow, L. G. de Lima, P. C. Dishuck, R. Durbin, T. Dvorkina, I. T. Fiddes, G. Formenti, R. S. Fulton, A. Fungtammasan, E. Garrison, P. G. S. Grady, T. A. Graves-Lindsay, I. M. Hall, N. F. Hansen, G. A. Hartley, M. Haukness, K. Howe, M. W. Hunkapiller, C. Jain, M. Jain, E. D. Jarvis, P. Kerpedjiev, M. Kirsche, M. Kolmogorov, J. Korlach, M. Kremitzki, H. Li, V. V. Maduro, T. Marschall, A. M. McCartney, J. McDaniel, D. E. Miller, J. C. Mullikin, E. W. Myers, N. D. Olson, B. Paten, P. Peluso, P. A. Pevzner, D. Porubsky, T. Potapova, E. I. Rogaev, J. A. Rosenfeld, S. L. Salzberg, V. A. Schneider, F. J. Sedlazeck, K. Shafin, C. J. Shew, A. Shumate, Y. Sims, A. F. A. Smit, D. C. Soto, I. é, J. M. Storer, A. Streets, B. A. Sullivan, F. Thibaud-Nissen, J. Torrance, J. Wagner, B. P. Walenz, A. Wenger, J. M. D. Wood, C. Xiao, S. M. Yan, A. C. Young, S. Zarate, U. Surti, R. C. McCoy, M. Y. Dennis, I. A. Alexandrov, J. L. Gerton, R. J. O’Neill, W. Timp, J. M. Zook, M. C. Schatz, E. E. Eichler, K. H. Miga, and A. M. Phillippy. The complete sequence of a human genome. *Science*, 376(6588):44–53, April 2022.
- 33 Sven Rahmann and Jens Zentgraf. Worst-case-optimal Spaced Seeds. Software, version 0.11., swbId: swb:1:dir:81ca043ed372e91711c1a9255974224264b1eb5d (visited on 2025-06-27). URL: <https://gitlab.com/rahmannlab/seed-optimization>, doi:10.4230/artifacts.23732.
- 34 M. Rautiainen and T. Marschall. GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biol*, 21(1):253, September 2020.
- 35 Luca Renders, Lore Depuydt, Sven Rahmann, and Jan Fostier. Automated design of efficient search schemes for lossless approximate pattern matching. In Jian Ma, editor, *Research in Computational Molecular Biology - 28th Annual International Conference, RECOMB 2024, Cambridge, MA, USA, April 29 – May 2, 2024, Proceedings*, volume 14758 of *Lecture Notes in Computer Science*, pages 164–184. Springer, 2024. doi:10.1007/978-1-0716-3989-4_11.
- 36 Luca Renders, Lore Depuydt, Sven Rahmann, and Jan Fostier. Lossless approximate pattern matching: Automated design of efficient search schemes. *J. Comput. Biol.*, 31(10):975–989, 2024. doi:10.1089/CMB.2024.0664.
- 37 K. Sahlin. Strobealign: flexible seed size enables ultra-fast and accurate read alignment. *Genome Biol*, 23(1):260, December 2022.
- 38 Valeriy Titarenko and Sofya Titarenko. Perfseeb: designing long high-weight single spaced seeds for full sensitivity alignment with a given number of mismatches. *BMC Bioinform.*, 24(1):396, 2023. doi:10.1186/S12859-023-05517-4.

- 39 Md. Vasimuddin, Sanchit Misra, Heng Li, and Srinivas Aluru. Efficient architecture-aware acceleration of BWA-MEM for multicore systems. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019*, pages 314–324. IEEE, 2019. doi:10.1109/IPDPS.2019.00041.
- 40 Jens Zentgraf and Sven Rahmann. Fast lightweight accurate xenograft sorting. *Algorithms Mol. Biol.*, 16(1):2, 2021. doi:10.1186/s13015-021-00181-w.
- 41 Jens Zentgraf and Sven Rahmann. Fast gapped k -mer counting with subdivided multi-way bucketed cuckoo hash tables. In Christina Boucher and Sven Rahmann, editors, *22nd International Workshop on Algorithms in Bioinformatics (WABI 2022)*, volume 242 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.WABI.2022.12.
- 42 Jens Zentgraf and Sven Rahmann. Swiftly identifying strongly unique k -mers. In Solon P. Pissis and Wing-Kin Sung, editors, *24th International Workshop on Algorithms in Bioinformatics, WABI 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 312 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.WABI.2024.15.
- 43 L. Zhang. Superiority of spaced seeds for homology search. *IEEE/ACM Trans Comput Biol Bioinform*, 4(3):496–505, 2007. doi:10.1109/TCBB.2007.1013.