frontiers in Genetics

METHODS
published: 25 June 2020
doi: 10.3389/fgene.2020.00572

Calibrating Seed-Based Heuristics to Map Short Reads With Sesame

Guillaume J. Filion [1,2*†], *Ruggero Cortini* [1] *and Eduard Zorita* [1]

[1] Center for Genomic Regulation (CRG), The Barcelona Institute of Science and Technology, Barcelona, Spain, [2] University Pompeu Fabra (UPF), Barcelona, Spain

The increasing throughput of DNA sequencing technologies creates a need for faster algorithms. The fate of most reads is to be mapped to a reference sequence, typically a genome. Modern mappers rely on heuristics to gain speed at a reasonable cost for accuracy. In the seeding heuristic, short matches between the reads and the genome are used to narrow the search to a set of candidate locations. Several seeding variants used in modern mappers show good empirical performance but they are difficult to calibrate or to optimize for lack of theoretical results. Here we develop a theory to estimate the probability that the correct location of a read is filtered out during seeding, resulting in mapping errors. We describe the properties of simple exact seeds, skip seeds and MEM seeds (Maximal Exact Match seeds). The main innovation of this work is to use concepts from analytic combinatorics to represent reads as abstract sequences, and to specify their generative function to estimate the probabilities of interest. We provide several algorithms, which together give a workable solution for the problem of calibrating seeding heuristics for short reads. We also provide a C implementation of these algorithms in a library called Sesame. These results can improve current mapping algorithms and lay the foundation of a general strategy to tackle sequence alignment problems. The Sesame library is open source and available for download at https://github.com/gui11aume/sesame.

Keywords: C library, probability, analytic combinatorics, seeding accuracy, heuristic algorithms

OPEN ACCESS

Edited by:
Max A. Alekseyev,
George Washington University,
United States

Reviewed by:
Debnath Pal,
Indian Institute of Science (IISc), India
Laurent Noe,
Université de Lille, France
Giovanni Manzini,
University of Eastern Piedmont, Italy

***Correspondence:**
Guillaume J. Filion
guillaume.filion@gmail.com

†Present address:
Guillaume J. Filion,
Department of Biological Sciences,
University of Toronto Scarborough,
Toronto, ON, Cananda

Specialty section:
This article was submitted to
Bioinformatics and Computational
Biology,
a section of the journal
Frontiers in Genetics

Received: *11 November 2019*
Accepted: *11 May 2020*
Published: *25 June 2020*

Citation:
Filion GJ, Cortini R and Zorita E (2020)
Calibrating Seed-Based Heuristics to
Map Short Reads With Sesame.
Front. Genet. 11:572.
doi: 10.3389/fgene.2020.00572

1. INTRODUCTION

1.1. Mapping Reads to Genomes

Say we use an imperfect instrument to sequence a small fragment of DNA. If we know its genome of origin, how can we find the location of the fragment in this genome?

The advent of high-throughput sequencing has put this question at the center of countless applications in genetics, such as discovering disease-causing mutations, selecting breeds of interest in agriculture or tracing human migrations.

We will refer to this as the *true location problem*. The answer to the problem depends on the length of the read, on the error rate of the sequencer and on one key feature of the genome: its repeat structure. Most genomes contain repetitive sequences, i.e., relatively large subsequences that are present at multiple locations. If the sequenced DNA fragment comes from a repetitive sequence, it may be impossible to map the read with certainty.

In general, repetitive sequences are not identical but merely homologous—meaning that their similarity is very unlikely to occur by chance. So if the sequenced DNA fragment originates from a repetitive sequence, we can map the read only if we can rule out all the candidates except one. This, in turn, depends on the similarity between the duplicates and on the error rate of the sequencer.

Since repetitive sequences play a central role in the problem, we give the terms "targets" and "duplicates" a meaning that will facilitate the exposition of the theory.

Definition 1. *The target is the DNA fragment that was actually sequenced. Duplicates are sequences of the genome that share homology with the target (in genetics they are often referred to as paralogs). In this article we will focus on short reads from complex eukaryotic genomes, so for concreteness the reader can assume that fragments are 30–300 bp long and that duplicates have above 75% identity with the target.*

The difficulty of the true location problem is due to sequencing errors. Occasionally, the sequence of a DNA fragment can be closer to one of the duplicates than to the target. So the *true* location of a DNA fragment is not always the *best* (as measured by the identity between the sequence and the candidate location). This naturally leads to asking how we can identify the *best* location of the fragment in the genome.

We will refer to this as the *best location problem*. It amounts to finding the optimal alignment between two sequences, and for this reason has received substantial attention in bioinformatics. For the purpose of developing a theory rooted in statistics, our main concern is to address the true location problem, but for simplicity, we will assume that the true location is also the best. When applicable, we will clarify the implications of this hypothesis in the relevant sections, and we will see how it impacts the theory as a whole in section 8.1.

1.2. Seeding Heuristics

Exact alignment algorithms were designed to solve the best location problem (Needleman and Wunsch, 1970; Smith and Waterman, 1981), but they are too slow to process the large amount of data generated by modern sequencers. Instead, one uses heuristic methods, i.e., algorithms that run faster but may return an incorrect result (Waterman, 1984).

The most popular heuristic for mapping DNA sequences is a filtration method called "seed-and-extend." The principle is to first identify seeds, defined as short regions of high similarity between the read and the genome, and then use an exact alignment algorithm at the seeded locations to evaluate the candidates and identify the optimum. Exact alignment algorithms, such as the Needleman-Wunsch (Needleman and Wunsch, 1970) or the Smith-Waterman (Smith and Waterman, 1981) algorithms return the distance or the similarity between two sequences according to some quantitative criterion. They are exact in the sense that they always return the correct answer, unlike heuristic algorithms. The seed-and-extend strategy was first proposed in FASTA (Lipman and Pearson, 1985) and BLAST (Altschul et al., 1990) to search for homology in sequence databases of proteins and DNA.

There are efficient methods to extract seeds, so it is possible to quickly hone in on a small set of candidates and to reduce the search space of the alignment algorithm. The disadvantage is that the target may not be in the candidate set, in which case the read cannot be mapped correctly.

As a consequence, seeding methods induce a trade-off between speed and accuracy: If the filtration is set to produce a large candidate set, the target is likely to be discovered, with the downside that checking all the candidates with an exact alignment algorithm will take long. Conversely, if the filtration is set to produce a small candidate set, the process will run faster but the target is more likely to be missed.

Importantly, the trade-off depends on the seeding method. This means that mapping algorithms can run faster at no cost for accuracy if we can find better seeding strategies. Progress on this line of research has largely benefited from the improvement of computer hardware and from the development of optimized data structures. There already exists a large body of literature on the design of seeding algorithms; the interested reader can find examples of those in references (Ma et al., 2002; Brejová, 2003; Li et al., 2004; Kucherov et al., 2005; Sun and Buhler, 2005; Xu et al., 2006; Lin et al., 2008). Sun and Buhler (2006) and Li and Homer (2010) present high-level comparisons of different designs, and Navarro (2001) gives a global overview of filtration methods in pattern matching.

1.3. The Two Types of Seeding Failure

Filtering heuristics are considered to fail whenever the target is not in the candidate set, but here we must be more specific and distinguish two kinds of failure: In the first kind, the candidate set contains a duplicate of the target but does not contain the target itself; in the second kind, the candidate set contains neither the target nor any duplicate.

The distinction is important because the duplicates of the target are similar to the read (due to their similarity to the target), so a failure of the first kind often looks like a success. In contrast, a failure of the second kind is easier to flag because in this case the candidates are not similar to the read. We will simply assume that seeding failures of the second kind are always detected (as explained in section 8.3), so that we can focus on the more difficult case of seeding failures of the first kind.

Before going further, we introduce three terms that will simplify the exposition.

Definition 2. *The output of the seeding step is the candidate set. The candidate set is the list of genomic locations where the read can be potentially mapped. The read is always mapped to one element of the candidate set. The seeding step is said to be*

(i) "on-target" if the candidate set contains the target,
(ii) "off-target" if the candidate set contains a duplicate but not the target,
(iii) "null" if the candidate set contains neither.

In this article, we will always consider that a genomic location is in the candidate set if and only if the read contains at least one seed with a perfect match for this genomic location.

With our assumptions, a read is mapped to the wrong location if and only if the candidate set is off-target. Indeed, if the candidate set is null, the read is not mapped, and if the candidate set is on-target, the correct location is discovered at the alignment step. The equivalence is granted by the assumption that the true location is also the best, reducing the mapping problem to a seeding problem. That being said, the true location is not always the best in practical applications. We will show in section 8.1 that off-target seeding can be responsible for most of the mapping errors even without the assumption above, but for now we maintain the strict equivalence between mapping errors and off-target seeding.

Focusing on popular heuristics to map short reads, our aim is to develop a method to estimate the probability that the candidate set is off-target. Previous work pioneered a method to compute seeding probabilities but it did not distinguish off-target from null seeding (Filion, 2017, 2018), and therefore did not provide a way to estimate the frequency of mapping errors. Other authors investigated the reliability of mapping algorithms (Menzel et al., 2013), but they focused on the probabilities of random hits, recognizing that addressing the problem of incorrect mapping requires taking into consideration the repeat structure of the genome.

The rest of the article is organized as follows: section 2 presents common seeding strategies used in bioinformatics, section 3 presents the basic concepts of analytic combinatorics that will be required to compute seeding probabilities, sections 4 to 6 treat the cases of exact seeds, skip seeds and MEM seeds, three common seeding heuristic used for mapping, section 7 presents Sesame, a C library implementing the main results of the theory, section 8 returns to the mapping problem and revisits the assumptions of the model, and finally section 9 provides some perspectives on the present work. **Appendix A** gathers for reference all the definitions encountered in the text, and **Appendix B** contains proofs and complements omitted from the main text.

2. SEEDS

The term "seed" has different meanings in computational biology. It can designate a part of the read, a part of the genome, a particular sequence motif, or a structured pattern of matches. Also, a seed does not always refer to an exact match. For instance, the algorithm PatternHunter (Ma et al., 2002) uses spaced seeds that tolerate mismatches. To avoid any confusion, we will adopt the convention below.

Definition 3. *A seed is a subsequence of the read that has size at least γ (defined by the context of the problem) and that has at least one perfect match in the reference genome. Every genomic match of every seed is in the candidate set.*

When a seed matches a given location of the genome, we say that it is a seed for that location. This is particularly useful in expressions such as "seed for the target" or "seed for a duplicate."

This definition presents a computational challenge: to know if a given subsequence of a read is a seed we need to know if it exists somewhere in the genome. This is a non trivial problem in itself, but fortunately we can use practical methods to solve it, even when the reference genome is very large.

These algorithms are crucial for the present theory, but describing them in depth is outside the scope of this document. Let us just mention that all the methods belong to a family known as exact offline string matching algorithms, where "offline" means that sequences are looked up in an index instead of the genome itself. Online methods can be used when the reference genome is not indexed (Faro and Lecroq, 2013), but this case is of little relevance in the present context.

The index is usually a hash table or a variant of the so-called FM-index (Ferragina and Manzini, 2000, 2005). Hash tables are typically used to index k-mers, which makes them useful to search for seeds of fixed size k (see Manekar and Sathe, 2018 for a recent benchmark of k-mer hashing algorithms). In contrast, some text-indexing structures have no set size so they can be used to search for seeds of different lengths. This is the case of the FM-index, a compact data structure based on the suffix array (Manber and Myers, 1993) and on the Burrows-Wheeler transform (Burrows and Wheeler, 1994), emulating a suffix trie with a much smaller memory footprint (Ferragina and Manzini, 2000, 2005).

Other methods can be efficient (e.g., running a bisection on the suffix array Dobin et al., 2013) but the FM-index is currently the most popular choice for seeding methods. For MEM seeds defined below, it is even the only practical option (Khan et al., 2009; Vyverman et al., 2013; Fernandes and Freitas, 2014; Khiste and Ilie, 2015). Overall, the detail is of little interest for the theory. We simply assume that seeds are known at all times without ambiguity because this problem has several practical solutions.

2.1. Exact Seeds

Exact seeds are seeds of fixed size γ. In other words, when using exact seeds, the candidate set consists of all the genomic locations for which there is a perfect match of size γ in the read. This seeding heuristic was used in the first version of BLASTN (Altschul et al., 1990), but it has become unpopular for producing many short spurious hits.

Figure 1 shows the exact seeds from an example read with three miscalled nucleotides. The sequenced DNA fragment has three duplicates so the seeds can match four possible locations.

Observe that erroneous nucleotides can belong to exact seeds because they sometimes match a duplicate. For instance, the first sequencing error matches all the duplicates and belongs to an off-target seed for the first duplicate. However, sequencing errors that are mismatches for *all* the sequences cannot belong to a seed. This is the case of the second sequencing error in the example, creating a local deficit of seeds.

Note the clutter in the middle of the read, where consecutive seeds match consecutive sequences at the same location. This is typical for exact seeds and is considered a nuisance for the implementation. Indeed, it is a waste of computer resources to discover matches in sequences that are already in the candidate set. In addition, this seeding method is not particularly sensitive compared to spaced seeds (Ma et al., 2002) so it is used only in a few specific applications. Nevertheless, it will be useful for the development of the present theory.

2.2. Skip Seeds

Skip seeds have a fixed size γ, but unlike exact seeds they cannot start at every nucleotide. Instead, a certain amount of nucleotides is skipped between every seed. This is a way to reduce the overlapping matches at the same location, at the cost of sensitivity. This seeding heuristic is the core of Bowtie2 (Langmead and Salzberg, 2012), where seeds have size $\gamma = 16$ and are separated by 10 nucleotides (nine positions are skipped). We will refer to seeds where n nucleotides are skipped as "skip-n seeds." For instance, Bowtie2 uses skip-9 seeds.

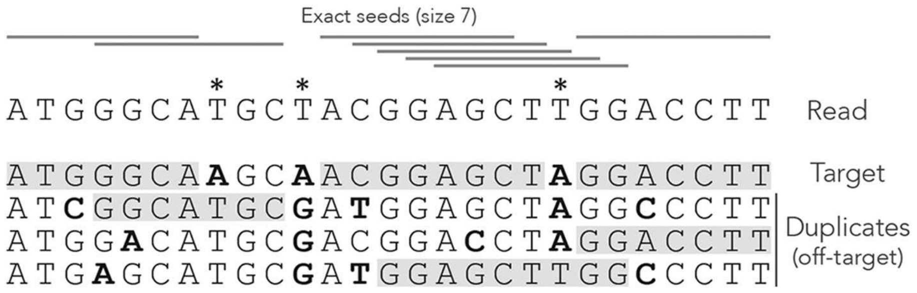

Exact seeds (size 7)
ATGGGCATGCTACGGAGCTTGGACCTT
Read
ATGGGCAAGCAACGGAGCTAGGACCTT
Target
ATCGGCATGCGATGGAGCTAGGCCCTT
ATGGACATGCGACGGACCTAGGACCTT
ATGAGCATGCGATGGAGCTTGGCCCTT
Duplicates
(off-target)

FIGURE 1 | Exact seeding. A sequenced DNA fragment (Read) is shown above the actual molecule (Target), and above three paralogs of the target (Duplicates). Sequencing errors are indicated by * (all three are A misread as T). The mismatches between the genomic sequences and the read are indicated in bold. The exact seeds of size $\gamma = 7$ are indicated as horizontal gray lines above the read. Matching regions in the genomic sequences are shadowed in gray. Several overlapping seeds accumulate at the center of the read, which is typical of exact seeds.

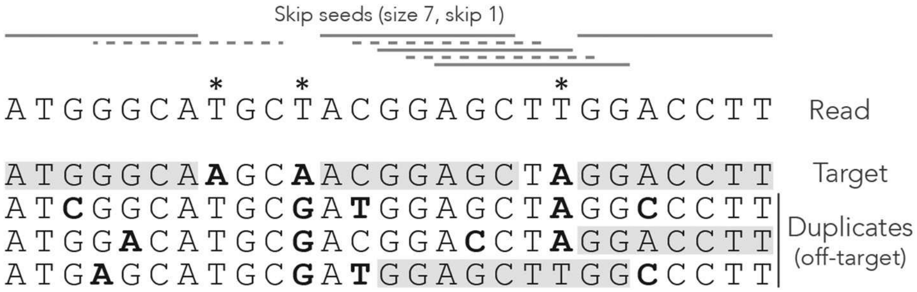

Skip seeds (size 7, skip 1)
ATGGGCATGCTACGGAGCTTGGACCTT
Read
ATGGGCAAGCAACGGAGCTAGGACCTT
Target
ATCGGCATGCGATGGAGCTAGGCCCTT
ATGGACATGCGACGGACCTAGGACCTT
ATGAGCATGCGATGGAGCTTGGCCCTT
Duplicates
(off-target)

FIGURE 2 | Example of skip seeds. The sequences and the annotations are the same as in **Figure 1**, but here we use skip-1 seeds. In other words, seeds can never start at nucleotides 2, 4, 6 etc. To highlight the difference with **Figure 1**, the missing seeds are represented by dotted lines.

Figure 2 shows what happens when exact seeds are replaced by skip-1 seeds on the read of **Figure 1**. Here the size is still $\gamma = 7$ but 1 nucleotide is skipped between seeds. This amounts to removing every second seed. The consequence is that there are fewer overlapping matches at the center of the read, but the only seed for the second duplicate is lost. This is a rather positive outcome because there is one off-target location fewer in the candidate set, but the same might happen to the target.

It is clear that skipping nucleotides reduces the sensitivity of the seeding step, but to what extent? One could test this empirically, but the answer depends on the seed length, the number of nucleotides that are skipped, the error rate of the sequencer and the size of the reads. The present theory will allow us to make general statements about the performance of skip seeds against exact seeds in different contexts.

2.3. MEM Seeds

MEM seeds (where MEM stands for Maximal Exact Match) are somewhat harder to define. Unlike exact seeds and skip seeds, their size is variable. They are used in BWA-MEM (Li, 2013) where they give good empirical results. To describe MEM seeds, let us first clarify the meaning of "Maximal Exact Match."

Definition 4. *A Maximal Exact Match (MEM) is a subsequence of the read that is present in the reference genome and that cannot be extended—either because the read ends or because the extended subsequence is not in the genome.*

A *MEM seed* is simply a MEM of size γ or greater. **Figure 3** shows what happens when using MEM seeds on the read of **Figure 1**. Observe that the clutter at the center of the read has disappeared because consecutive matches are fused into a few MEM seeds.

Two consecutive MEM seeds can overlap, in which case they always match distinct sequences of the genome (otherwise neither of them would be a MEM seed). There does not have to be any overlap though, because a nucleotide can be a mismatch against *all* the sequences, like the second read error for instance.

Note that a MEM does not always match a single subsequence of the genome. For instance, the rightmost MEM seed matches two distinct genomic subsequences. This case motivates the following definition, which will play an important role later.

Definition 5. *A strict MEM seed has a single match in the genome. A shared MEM seed has several matches in the genome.*

Compared to seeds of fixed size, MEM seeds have two counter-intuitive properties. The first is that there are cases where there cannot be any on-target seed, even when changing the minimum seed size γ. **Figure 4** shows such an example. Even though there is a single sequencing error, the read has no MEM seed for the target. Lowering γ does not change this, so there is no way to discover the true location using MEM seeds (even though it is the best location).

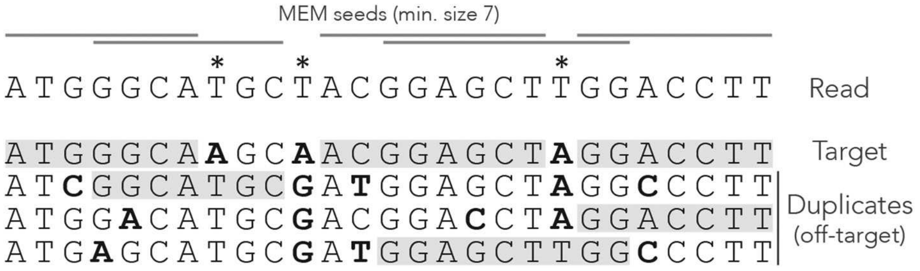

MEM seeds (min. size 7)
ATGGGCATGCTACGGAGCTTGGACCTT Read
ATGGGCAAGCAACGGAGCTAGGACCTT Target
ATCGGCATGCGATGGAGCTAGGCCCTT
ATGGACATGCGACGGACCTAGGACCTT
ATGAGCATGCGATGGAGCTTGGCCCTT
Duplicates (off-target)

FIGURE 3 | Example of MEM seeds. The sequences and the annotations are the same as in **Figure 1**, but here we use MEM seeds of minimum size $\gamma = 7$. The clutter at the center of the read has disappeared and there is at least one seed for each sequence.

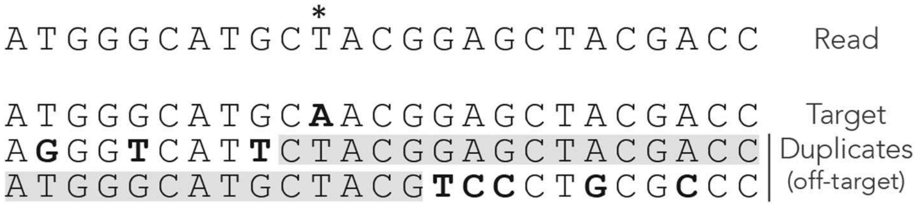

ATGGGCATGCTACGGAGCTACGACC Read
ATGGGCATGCAACGGAGCTACGACC Target
AGGGTCATTCTACGGAGCTACGACC
ATGGGCATGCTACGTCCCTGCGCCC
Duplicates (off-target)

FIGURE 4 | Issues with MEM seeds: inaccessible targets. The read, the MEM seeds and the sequences are represented as in **Figure 3**. The MEM seeds matching the two duplicates at the bottom effectively hide the target, so it cannot be discovered. This can occur even when the true location is the best candidate and when there is a single error on the read.

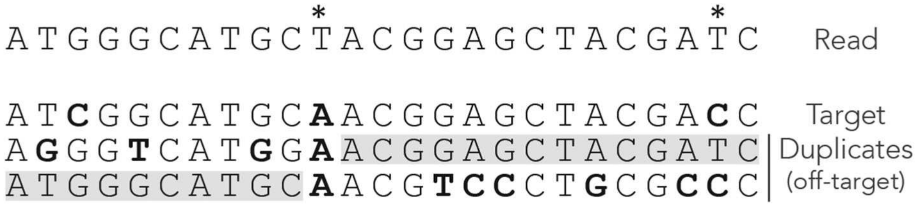

ATGGGCATGCTACGGAGCTACGATC Read
ATCGGCATGCAACGGAGCTACGACC Target
AGGGTCATGGAACGGAGCTACGATC
ATGGGCATGCAACGTCCCTGCGCCC
Duplicates (off-target)

FIGURE 5 | Issues with MEM seeds: too long reads. The read, the MEM seeds, and the sequences are represented as in **Figure 3**. There would be an on-target seed (shared) if the read were two nucleotides shorter. The true location is hidden by the last two nucleotides.

The second counter-intuitive property is that shortening a read can sometimes generate a MEM seed for the target. **Figure 5** shows an example of this case. There is no MEM seed for the target, but there would be if the read were two nucleotides shorter on the right side. Indeed, in this case there would be a shared MEM seed matching the target and the first duplicate (provided $\gamma \leq 12$).

These examples show that MEM seeds can perform worse than seeds of fixed size. MEM seeds yield fewer candidates and therefore speed up the mapping process, but the question is at what cost? The theory developed here will allow us to compute the probability that a read has no MEM seed for the target and thus that the true location is missed at the seeding stage.

2.4. Spaced Seeds

Originally introduced by Califano and Rigoutsos (1993) and popularized by PatternHunter (Ma et al., 2002), spaced seeds feature "don't-care" positions allowing them to detect imperfect matches. Spaced seeds are represented by models such as "11*11," here indicating that the seeds have length 5 and that the middle position is disregarded. At index time, the genome is scanned with the model so that nucleotides labeled "1" are concatenated and indexed. At search time, the query is scanned with the model and the concatenated nucleotides labeled "1" are looked up in the index.

Spaced seeds of weight w (the number of 1 in the model) have the same memory requirements as contiguous seeds of length w but they are more sensitive (Li et al., 2006), making them very attractive for homology search. They were used in the first generation of short-read mappers (Jocham et al., 1986; Lin et al., 2008; Chen et al., 2009; Rumble et al., 2009), but nowadays they find more applications in genome comparisons (Kiełbasa et al., 2011; Healy, 2016), metagenomics (Břinda et al., 2015; Ounit and Lonardi, 2016), genome assembly (Birol et al., 2015), and long-read mapping (Sovic et al., 2016).

The success of the mainstream mappers BWA-MEM (Li, 2013) and Bowtie2 (Langmead and Salzberg, 2012) is due in part to the FM-index, which only supports contiguous seeds.

Some workarounds are available for spaced seeds (Horton et al., 2008; Gagie et al., 2017) but they increase the memory footprint, explaining that short reads are typically mapped using contiguous seeds. More generally, computing the sensitivity of spaced seeds is challenging (Kucherov et al., 2006; Li et al., 2006; Martin and Noé, 2017). It is possible to do this using the tools introduced below, as shown in section 3.5. However, to compute off-target probabilities, the main purpose of this article, the complexity rapidly becomes prohibitive. We will thus restrict our attention to contiguous seeds because they are relevant for mapping problems and fit tightly within the present theory.

3. MODEL AND STRATEGY

3.1. Sequencing Errors and Divergence of the Duplicates

We now need to model the sequencing and duplication processes so that we can compute the probabilities of the events of interest. We assume that the sequencing instrument has a constant substitution rate p, and that insertions and deletions never occur. When a substitution occurs, we assume that the instrument is equally likely to output any one of the remaining three nucleotides. This corresponds more or less to the error spectrum of the Illumina sequencing technology (Nakamura et al., 2011).

Next, we assume that the target sequence has $N \geq 0$ duplicates, so that there are N off-target sequences. We further assume that duplication happened instantaneously at some point in the past and that all $N + 1$ sequences diverge independently of each other at a constant rate. In other words, we ignore the complications due to the genealogy of the duplication events. Instead, we simply assume that at each nucleotide position, any given duplicate is identical to the target with probability $1 - \mu$. If it is not, we assume that the duplicate sequence can be any of one the three remaining nucleotides (i.e., each is found with probability $\mu/3$).

Note that read errors are always mismatches against the target (because we assume that the target is the true sequence), and they match each duplicate with probability $\mu/3$. Correct nucleotides are always matches for the target, and they match each duplicate with probability $1 - \mu$.

Before going further, we also need to move a practical consideration out of the way. Seeds can match any sequence of the genome, not just the target or the duplicates. However, we will ignore matches in the rest of the genome because such random matches are unlikely to cause a mapping error when seeding is off-target, contrary to matches in duplicates. Neglecting those will greatly simplify the exposition of the theory without loss of generality. We will explain in section 8.3 how to deal with this practical case and how to identify those matches as off-target. Until then, we will consider that the target and the duplicates are the only sequences in the reference genome.

3.2. Weighted Generating Functions

The central object of analytic combinatorics is the generating function, and for our purpose we will use a special kind known as *weighted generating function*.

Definition 6. *Let $\mathcal{A}$ be a set of combinatorial objects such that $a \in \mathcal{A}$ has a size $|a| \in \mathbb{N}$ and a weight $w(a) \in \mathbb{R}^+$. The weighted generating function of $\mathcal{A}$ is defined as*

$$A(z) = \sum_{a \in \mathcal{A}} w(a) z^{|a|}, \tag{1}$$

Expression (1) also defines a sequence $(a_k)_{k \geq 0}$ such that

$$A(z) = \sum_{k=0}^{\infty} a_k z^k.$$

By definition $a_k = \sum_{a \in A_k} w(a)$, where A_k is the class of objects of size k in $\mathcal{A}$. The number a_k is the total weight of objects of size k.

To give an example, assume that a particular symbol, say $\Downarrow$, has a probability of occurrence equal to p. The weighted generating function of words containing only this symbol is pz. The weight of the word is its probability (here equal to p) and the size is its length (here 1).

In this document we will focus on the weighted generating function $A(z)$ of the set $\mathcal{A}$ of reads that do not have on-target seeds (i.e., reads for which seeding is either null or off-target). The weight of a read is its probability of occurrence and the size k is its number of nucleotides. The coefficient a_k is thus the proportion of reads of size k that do not have an on-target seed, which is the quantity of interest.

The motivation for introducing weighted generating functions is that operations on combinatorial objects translate into operations on their weighted generating functions. If $A(z)$ and $B(z)$ are the weighted generating functions of two mutually exclusive sets $\mathcal{A}$ and $\mathcal{B}$, the weighted generating function of $\mathcal{A} \cup \mathcal{B}$ is $A(z) + B(z)$, as evident from expression (1). Size and weight can be defined for pairs of objects in $\mathcal{A} \times \mathcal{B}$ as $|(a, b)| = |a| + |b|$ and $w(a, b) = w(a)w(b)$. In other words the sizes are added and the weights are multiplied. With this convention, the weighted generating function of the Cartesian product $\mathcal{A} \times \mathcal{B}$ is $A(z)B(z)$. This simply follows from expression (1) and from

$$A(z)B(z) = \sum_{a \in \mathcal{A}} w(a) z^{|a|} \sum_{b \in \mathcal{B}} w(b) z^{|b|} = \sum_{(a,b) \in \mathcal{A} \times \mathcal{B}} w(a) w(b) z^{|a|+|b|}.$$

These two operations are all we need in order to compute the weighted generating functions of the reads of interest. Addition corresponds to creating a new family by merging reads from families $\mathcal{A}$ and $\mathcal{B}$. Multiplication corresponds to creating a new family by concatenating reads from families $\mathcal{A}$ and $\mathcal{B}$.

3.3. Analytic Representation

The analytic combinatorics framework relies on a strategy referred to as the *symbolic method* (Sedgewick and Flajolet, 2013). The idea is to combine simple objects into more complex objects. Each combinatorial operation on the objects corresponds to a mathematical operation on their weighted generating functions. One can thus obtain the weighted generating function of complex objects, whose coefficients a_k ($k \geq 0$) are the probabilities of interest (Régnier, 2000; Nicodeme et al., 2002; Sedgewick and Flajolet, 2013).

As explained by Filion (2017, 2018), we recode the reads in alphabets of custom symbols and we specify a construction plan of the reads using a so-called *transfer matrix* $M(z)$. The transfer matrix specifies which types of "segments" can follow each other in the reads of interest: the entry at coordinates (i, j) is the weighted generating function of segments of type j that can be appended to segments of type i.

$M(z)$ contains the weighted generating functions of all the reads that consist of a single segment. From the basic operations on weighted generating functions, $M(z)^s$ contains the weighted generating functions of all the reads that consist of s segments. Therefore, the entry at coordinates (i, j) of the matrix $M(z) + M(z)^2 + M(z)^3 + \ldots = M(z) \cdot (I - M(z))^{-1}$ is the weighted generating function of the reads of any size and any number of segments, that end with a segment of type j and that can be appended to a segment of type i. The examples below will clarify the key steps of this strategy.

A complete description of how to compute seeding probabilities with the symbolic method is given by Filion (2017, 2018). The interested readers can also find more about analytic combinatorics in popular textbooks (Flajolet and Sedgewick, 2009; Sedgewick and Flajolet, 2013).

3.4. Example 1: On-Target Exact Seeds

We highlight the strategy above with an example that will turn out to be central for the development of the theory. In addition, it is simple enough to provide a gentle introduction to the general methodology. This example was described in detail by Filion (2017, 2018), but we repeat it here with a different formalism to fit the present article.

The first step is to note that the nucleotide sequence of the read is irrelevant. Indeed, the read has an on-target seed if and only if it contains a stretch of γ nucleotides without error. For this reason, we recode reads as sequences of correct or erroneous nucleotides.

We define the mismatch alphabet $\mathcal{A}_0 = \{\square, |, \Downarrow\}$, where $\square$ represents a correct nucleotide, $\Downarrow$ represents an erroneous nucleotide and $|$ is a special symbol appended after the last nucleotide to mark the end of the read. It is not associated with a nucleotide and therefore has size 0.

This recoding allows us to partition the read in an important way.

Definition 7. *A terminator is any symbol that is different from the symbol* $\square$. *A segment is a sequence of 0 or more* $\square$ *symbols followed by a terminator. The tail is the last segment of the read, where the terminator is always the special symbol* $|$.

Since the decomposition of the read in segments is unique, we can view a read as a sequence of segments with a tail, instead of a sequence of nucleotides. **Figure 6** shows an example of decomposition in segments. On-target seeds cannot contain sequencing errors, therefore they must be completely embedded in a segment. So the sizes of the segments indicate whether the read contains an on-target seed or not.

The probability of occurrence of the symbol $\Downarrow$ is p (the error rate of the sequencer) so the probability of occurrence of the $\square$ symbol is $1 - p = q$. Both symbols have size 1, so their respective weighted generating functions are pz and qz. Using the rule for concatenation, we see that a segment of i symbols $\square$ followed by a terminator has weighted generating function $(qz)^i pz$. The final symbol $|$ has size 0, so a tail segment of i symbols $\square$ followed by the symbol $|$ has weighted generating function $(qz)^i$.

The key insight is that the reads without on-target seed are exactly the reads that are made of segments with fewer than γ symbols $\square$, where γ is the minimum seed size. The weighted generating function of such segments is $\big(1 + qz + \ldots + (qz)^{\gamma-1}\big)pz$, and that of the tail is $1 + qz + \ldots + (qz)^{\gamma-1}$. This gives a construction plan that can be encoded in a transfer matrix.

Reads consist of only two kinds of objects: the segments terminated by $\Downarrow$ (or $\Downarrow$-segments for short) and the tails, so the dimension of the transfer matrix is 2×2. A $\Downarrow$-segment can be followed by another $\Downarrow$-segment or by the tail. The tail cannot be followed by anything. The expression of the transfer matrix $M_0(z)$ is thus

$$\begin{array}{cc} & \begin{array}{cc} \quad\quad\quad\Downarrow & \quad\quad\quad\quad\quad\quad | \end{array} \\ \begin{array}{c} \Downarrow \\ | \end{array} & \begin{bmatrix} \big(1 + qz + \ldots + (qz)^{\gamma-1}\big)pz & 1 + qz + \ldots + (qz)^{\gamma-1} \\ 0 & 0 \end{bmatrix} \end{array},$$

where p is the error rate of the sequencer, $q = 1 - p$ and γ is the minimum seed length. In the representation above, the different types of segments are identified by their terminator, indicated in the margins for clarity.

The entries of $M_0(z)$ correspond to sequences of one segment with no seed. Likewise, the entries of $M_0(z)^s$ correspond to sequences of s segments with no seed and the entries of $M_0(z) + M_0(z)^2 + \ldots = M_0(z) \cdot (I - M_0(z))^{-1}$ correspond to sequences

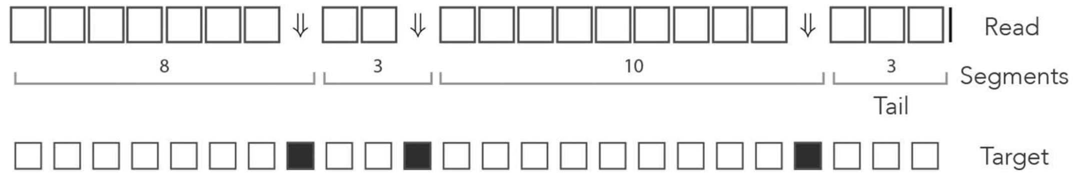


FIGURE 6 | The mismatch encoding. An example read is represented in the mismatch alphabet. The symbol $\Downarrow$ represents a mismatch against the target (an erroneous nucleotide) and the symbol $\square$ represents a match (a correct nucleotide). The symbol $|$ is appended to the end of the read. The symbolic sequence of the target is represented below, where an open square stands for a match and a closed square stands for a mismatch.

of any number of segments with no seed. The entry of interest is the top right term, associated with terminators ⇓ and |. To see why, observe that every read can be prepended by ⇓-segments and only by those (not by a tail). Thus, reads are precisely the sequences of segments that can follow the symbol ⇓ and that are terminated by a tail, whose weighted generating function is the top right entry of the matrix. In **Appendix B.1**, we show that this term is equal to

$$\frac{1 + qz + \ldots + (qz)^{\gamma-1}}{1 - \left(1 + qz + \ldots + (qz)^{\gamma-1}\right)pz}. \tag{2}$$

This function can be expanded as a Taylor series $a_0 + a_1 z + a_2 z^2 + \ldots$, but the coefficients $a_0, a_1, a_2, \ldots$ are unknown. By construction, a_k is the quantity of interest, i.e., it is the probability that a read of size k does not contain an on-target seed of minimum size γ, so we now need to extract the Taylor coefficients from expression (2). There are several methods to do so; the one we choose here is to build a recurrence equation. In **Appendix B.2**, we show that

$$a_k = \begin{cases} 1 & \text{if } k < \gamma, \\ 1 - q^\gamma & \text{if } k = \gamma, \\ a_{k-1} - pq^\gamma \cdot a_{k-\gamma-1} & \text{otherwise.} \end{cases} \tag{3}$$

Note that for $k < \gamma$, the read is too short so the probability that it contains no seed is 1; for $k = \gamma$, the read contains a seed if and only if it has no error, which occurs with probability q^γ.

The terms of interest can be computed recursively using expression (3). This approach is very efficient because every iteration involves at most one multiplication and one subtraction. Also, the default floating-point arithmetic on modern computers gives sufficient precision to not worry about numeric instability for the problems considered here (we rarely need to compute those probabilities for reads above 500 nucleotides).

This example shows how the symbolic approach yields a non-trivial and yet simple algorithm to compute the probability that a read of size k does not contain an on-target exact seed.

3.5. Example 2: On-Target Spaced Seeds

The goal of our second example is to exhibit the mechanisms of our strategy in a more complex setting. In this example we propose to compute the probability that a read of size k contains a match for the spaced seed "11*1*1." This problem has no concrete application, but it illustrates in a relatively simple way the general methodology to deal with spaced seeds.

Here we proceed exactly as in the previous section, replacing nucleotides by the symbols from the mismatch alphabet $\mathcal{A}_0 = \{\Box, |, \Downarrow\}$. Recall that □ represents a correct nucleotide, ⇓ represents an erroneous nucleotide and | is the special terminator that marks the end of the read. As explained in the previous example, the weighted generating functions of the symbols □, ⇓ and | are qz, pz, and 1, respectively, where p is the probability of a sequencing error and $q = 1 - p$. We again decompose reads into unique sequences of segments, where a segment is the concatenation of zero or more □ symbols with a terminator ⇓, or | for the last segment also called the tail.

So far everything is identical to the previous example, the difference is how we characterize reads that do not contain a seed. This is where the transfer matrix comes in handy. We define four abstract states that represent how the end of the segment matches the seed model "11*1*1." The states are represented as ⇓, □□ ⇓, □□ ⇓ □ ⇓, and □□□□ ⇓. All the states finish with ⇓ because they correspond to the end of one or more internal segments, which are always terminated by ⇓. To these states we add | for the tail.

To fill the transfer matrix, we need to indicate how the segments bring the read from a state to another. We illustrate how this is done with the transitions from state ⇓. This state indicates that the longest alignment of the previous segment with the seed model has zero nucleotide. If the next segment is ⇓ or □ ⇓, the longest alignment with the seed model will again have length 0 and the state will remain ⇓. If the next segment is □□ ⇓ or □□□ ⇓, the last three nucleotides align with the beginning of the seed model "11*," bringing the read to the state □□ ⇓. If the next segment is □□□□ ⇓ or □□□□□ ⇓, the last five nucleotides align with "11*1*," bringing the read to the state □□□□ ⇓. Finally, segments with more than five □ symbols are disallowed because they create a match for the seed. In all of these segments, the ⇓ terminator can be replaced by | to indicate the end of the read.

Recall from the previous section that the weighted generating function of a segment with i symbols □ and a terminator ⇓ is $(qz)^i pz$, and that the weighted generating function of a tail with i symbols □ is $(qz)^i$. With this information we can fill the row of the transfer matrix that corresponds to state ⇓. With the same logic, we can also fill the other entries of the transfer matrix, making sure that we exclude all the possible matches with the seed. Doing this term by term, we obtain the transfer matrix $M_\diamond(z)$ equal to

$$\begin{array}{c} \\ \Downarrow \\ \Box\Box\Downarrow \\ \Box\Box\Downarrow\Box\Downarrow \\ \Box\Box\Box\Box\Downarrow \\ | \end{array} \begin{array}{c} \begin{array}{ccccc} \Downarrow & \Box\Box\Downarrow & \Box\Box\Downarrow\Box\Downarrow & \Box\Box\Box\Box\Downarrow & | \end{array} \\ \begin{bmatrix} (1+qz)pz & \left((qz)^2 + (qz)^3\right)pz & 0 & \left((qz)^4 + (qz)^5\right)pz & F_5(z) \\ pz & (qz)^2 pz & qzpz & 0 & F_2(z) \\ pz & 0 & 0 & 0 & F_0(z) \\ pz & 0 & 0 & 0 & F_0(z) \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \end{array}$$

where

$$F_i(z) = 1 + qz + \ldots + (qz)^i.$$

The entries of $M_\diamond(z) + M_\diamond(z)^2 + \ldots = M_\diamond(z) \cdot (I - M_\diamond(z))^{-1}$ correspond to sequences of any number of segments with no match for the seed. The entry of interest is the top right term, associated with states ⇓ and |. To see why, observe that every read can be prepended by segments finishing in state $\Downarrow$ and only by those, otherwise the read would already match some part of the seed before the first nucleotide.

The matrix $M_\diamond(z)$ is simple enough that $M_\diamond(z) \cdot (I - M_\diamond(z))^{-1}$ can be computed explicitly using a computer algebra system. The weighted generating function of interest is found to be the ratio of two polynomials $P(z)/Q(z)$ where P has degree 9 and Q has degree 10. From there, one can proceed as in **Appendix B.2** to obtain a recurrence of order 10 that gives the coefficient a_k in the Taylor expansion of $P(z)/Q(z)$. By construction, this coefficient a_k is the probability that a read of length k does not contain any match for the spaced seed "11*1*1." Alternatively, one can compute a_k directly from the powers of $M_\diamond(z)$ as explained in the next section.

This approach can be applied for any spaced seed. A seed model with m don't-care positions generates 2^m possible states and the dimension of the associated transfer matrix is $2^m + 1$ (one state is reserved for the tail). For large m it is impossible to compute $(I - M_\diamond(z))^{-1}$ analytically but one can still compute the powers of $M_\diamond(z)$ efficiently because the matrix is sparse. So in general, the computational method introduced in the next section is more adapted.

3.6. Example 3: On-Target Skip Seeds

Let us now go through an example that will be important later. Here we devise a method to compute the probability that a read contains no on-target skip seed. Using the same strategy as in the previous example, we start by recoding the reads using a specialized alphabet to solve this problem.

We need to know whether a nucleotide is a sequencing error, but this time we also need to know its phase in the repeated cycles of skipped positions. For this, we define the skip-n alphabet $\mathcal{A}_n = \{\square, |, \Downarrow_0, \Downarrow_1, \ldots, \Downarrow_n\}$. Again, the symbol $\square$ represents a correct nucleotide and the symbol $|$ is a terminator added at the end of the read. The symbols $\Downarrow_j$ $(0 \leq j \leq n)$ represent sequencing errors and j indicates the number of nucleotides until the next non-skipped position (i.e., $j = 0$ for nucleotides immediately before a non-skipped position and $j = n$ for nucleotides at a non-skipped position).

As per definition 7, segments in this alphabet are sequences of 0 or more $\square$ symbols followed by any of the symbols $\Downarrow_j$ or by the symbol $|$. Given that this decomposition is unique, we can again view a read as a sequence of segments with a tail. The example of **Figure 6** is shown again in **Figure 7**, where segments in the mismatch alphabet have been replaced by segments in the skip-3 alphabet.

The probability of occurrence of a sequencing error is p, so every symbol $\Downarrow_j$ has the same weighted generating function pz—provided the next non-skipped position is at distance j, otherwise the weighted generating function is 0. The weighted generating function of the symbol $\square$ is again qz, so a segment with i symbols $\square$ followed by the symbol $\Downarrow_j$ has weighted generating function $(qz)^i pz$—if the next non-skipped position is at distance j, otherwise it is 0. As in the previous example, a tail segment with i symbols $\square$ followed by the symbol $|$ has weighted generating function $(qz)^i$.

This case is more complex than the previous one: reads without on-target skip seed of minimum size γ can contain segments with γ or more $\square$ symbols. For instance, the read shown in **Figure 7** contains a stretch of 9 nucleotides without errors but it has no seed of minimum size $\gamma = 9$. More generally, if there is a sequencing error i nucleotides before the next non-skipped position, there can be up to $\gamma + i - 1$ symbols $\square$ in a row.

Expressed in different words, it is possible to append segments with up to $\gamma + i - 1$ symbols $\square$ after segments terminated by $\Downarrow_i$ ($\Downarrow_i$-segments for short). Each of those $\gamma + i$ possible segments is associated with a different terminator, depending on how far ahead the next non-skipped position lies. In **Figure 7**, for instance, the second segment is terminated by $\Downarrow_1$ because there is 1 nucleotide before the next non-skipped position. If the segment were 1 nucleotide longer, the terminator would have to be $\Downarrow_0$.

The main issue is that segments of different lengths can be terminated by the same symbol. Going back to **Figure 7**, the third segment has length 10 and is terminated by $\Downarrow_3$. It would also be terminated by $\Downarrow_3$ if it had length 2 or length 6. In the general case, the entries of the transfer matrix show some periodicity modulo $n + 1$.

Denote $H_{i,j}(z)$ the weighted generating function of $\Downarrow_j$-segments that can follow a $\Downarrow_i$-segment. The total number of segments that can follow a $\Downarrow_i$-segment is $\gamma + i$. Among them, the shortest $\Downarrow_j$-segment has size ℓ_0 to be determined below, and the others have sizes $\ell_0 + (n+1), \ell_0 + 2(n+1), \ldots, \ell_0 + m(n+1)$, for some integer m. This m is the largest number such that $\ell_0 + m(n+1) \leq \gamma + i$, so $m = \lfloor (\gamma + i - \ell_0)/(n+1) \rfloor$, where $\lfloor \ldots \rfloor$ is the "floor" function.

Those segments follow a $\Downarrow_i$-segment, so they start i nucleotides before the next non-skipped position. The shortest following segment that ends j nucleotides before a non-skipped position has length $\ell_0 = i - j$ if $i > j$, and $n + 1 - j + i$ otherwise. This is equivalent to defining ℓ_0 as $x + 1$ where x is the number of $\square$ symbols of the shortest $\Downarrow_j$-segment, i.e., $x = i - j - 1 \pmod{n+1}$.

With these notations, the shortest $\Downarrow_j$-segment consists of x symbols $\square$ followed by the $\Downarrow_j$ terminator, so it has weighted generating function $(qz)^x pz$. The other segments have a multiple of $n + 1$ extra $\square$ symbols so their weighted generating functions are $(qz)^{x+(n+1)}pz, (qz)^{x+2(n+1)}pz, \ldots, (qz)^{x+m(n+1)}pz$. Summing over those cases, we finally obtain

$$H_{i,j}(z) = (qz)^x \big(1 + (qz)^{n+1} + \ldots + (qz)^{m(n+1)}\big) pz,$$
$$\text{where } x = i - j - 1 \pmod{n+1}, \text{ and}$$
$$m = \left\lfloor \frac{\gamma + i - 1 - x}{n+1} \right\rfloor. \tag{4}$$

Denote $J_i(z)$ the weighted generating function of tail segments that can follow a $\Downarrow_i$-segment. There are $\gamma + i$ such tails, each consisting of 0 to $\gamma + i - 1$ symbols $\square$ followed by the special $|$ terminator for the end of the read. The $|$ symbol has size 0 so its weighted generating function is 1. Once again summing over the different cases, we obtain

$$J_i(z) = 1 + qz + (qz)^2 + \ldots + (qz)^{\gamma+i-1}. \tag{5}$$

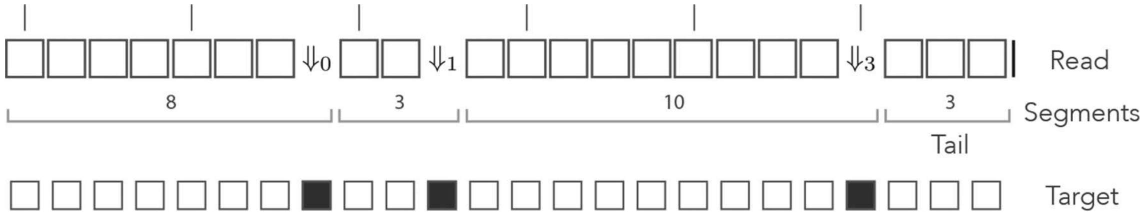


FIGURE 7 | The skip encoding. The read of **Figure 6** is represented in the skip-3 alphabet. The symbols $\Downarrow_j$ ($j = 1, 2, 3$) represent mismatches against the target (they are erroneous nucleotides) and the □ symbol represents a match (it is a correct nucleotide). The vertical bars indicate non-skipped positions (the potential start of a seed). The number j in $\Downarrow_j$ indicates the number of nucleotides until the next non-skipped position. For $j = 0$ the next nucleotide is not skipped. Other features are as in **Figure 6**.

Finally, the expression of $M_n(z)$ the transfer matrix of reads in the skip-n alphabet is

$$\begin{array}{cc} & \begin{array}{ccccc} \Downarrow_0 & \Downarrow_1 & \dots & \Downarrow_n & | \end{array} \\ \begin{array}{c} \Downarrow_0 \\ \Downarrow_1 \\ \vdots \\ \Downarrow_n \\ | \end{array} & \begin{bmatrix} H_{0,0}(z) & H_{0,1}(z) & \dots & H_{0,n}(z) & J_0(z) \\ H_{1,0}(z) & H_{1,1}(z) & \dots & H_{1,n}(z) & J_1(z) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ H_{n,0}(z) & H_{n,1}(z) & \dots & H_{n,n}(z) & J_n(z) \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \end{array}, \qquad (6)$$

where p is the error rate of the sequencer, $q = 1 - p$, n is the number of skipped nucleotides between potential seeds, γ is the minimum seed length, and polynomials $H_{i,j}(z)$ and $J_i(z)$ are defined in expressions (4) and (5).

In **Appendix B.3**, we present a transfer matrix with a simpler expression that will prove useful in section 5. The version of **Appendix B.3** is simpler, but the expression above has some advantages that will be explained below.

The weighted generating function of interest is the top right entry of the matrix $M_n(z) + M_n(z)^2 + \ldots = M_n(z) \cdot (I - M_n(z))^{-1}$. To see why, observe that, at the start of every read, the next nucleotide is a non-skipped position, so every read can be prepended by $\Downarrow_0$-segments and only by those. Thus, reads are precisely the sequences of segments that can follow the symbol $\Downarrow_0$ and that are terminated by a tail, whose weighted generating function is the entry of the matrix associated with terminators $\Downarrow_0$ and |.

By construction, the Taylor expansion of the top right term in the matrix $M_n(z) \cdot (I - M_n(z))^{-1}$ contains the probabilities of interest. More specifically, if the Taylor series of this term is $a_0 + a_1 z + a_2 z^2 + \ldots$, then a_k is the probability that a read of size k contains no skip-n seed of minimum size γ.

But $M_n(z)$ is too complex to find a closed expression for $M_n(z) \cdot (I - M_n(z))^{-1}$ or its top right term. Instead, we return to the definition $M_n(z) + M_n(z)^2 + \ldots$ and show in **Appendix B.4** that the coefficient of interest, a_k, only depends on the first $k + 1$ terms of the sum. So for reads of size k or lower, we only need to compute the matrix $M_n(z) + M_n(z)^2 + \ldots + M_n(z)^{k+1}$ and work out the Taylor expansion of the top right term.

But we can do better: since we are only interested in the coefficients up to order k, we can perform all algebraic operations on truncated polynomials of order k, i.e., we discard the coefficients of order $k + 1$ or greater when multiplying two polynomials.

But we can do even better: a read with $s + 1$ segments contains s errors, so the top right entry of $M_n(z)^{s+1}$ is the weighted generating function of reads with s errors that have no seed of minimum size γ. Computing the partial sum $M_n(z) + M_n(z)^2 + \ldots + M_n(z)^s$ instead of $M_n(z) + M_n(z)^2 + \ldots + M_n(z)^{k+1}$ corresponds to neglecting reads with s or more errors. For s sufficiently large, such reads are exceedingly rare so we can obtain accurate estimates without computing all the powers of $M_n(z)$ up to order k.

The number of errors X in a read of size k has a Binomial distribution $X \sim B(k, p)$. From Arratia and Gordon (1989) we can bound the probabilities of the tail with the expression

$$Pr(X \geq s) \leq \exp\left((s - k)\log\frac{k - s}{k(1 - p)} - s\log\frac{s}{kp}\right). \qquad (7)$$

Using the formula above, we can thus bound the probability that a read has $s+1$ or more segments. We compute $M_n(z) + M_n(z)^2 + \ldots + M_n(z)^{s+1}$ where the weighted generating functions have been replaced by truncated polynomials and we extract the top right entry. When the right-hand side of expression (7) is lower than a set fraction ε of the current value of a_k, we stop the computations. Typically $\varepsilon = 0.01$ so this method ensures that the probabilities that a read of size k has no on-target skip seed are accurate to within 1%.

With $M_n^*(z)$, the transfer matrix of **Appendix B.3**, the top right entry of $M_n^*(z)^{s+1}$ is not the weighted generating function of reads with s errors, so (7) is not an upper bound for the neglected terms of the sum. As a consequence, one would have to compute more terms in the partial sum $M_n^*(z) + M_n^*(z)^2 + \ldots + M_n^*(z)^{s+1}$ to reach the same accuracy. The transfer matrix shown in expression (6) is not the simplest, but it has the benefit of requiring fewer iterations.

Remark 1. *Observe that when $n = 0$ the matrix $M_n(z)$ is identical to the matrix $M_0(z)$ of section 3.4. This is consistent with the fact that exact seeds are skip-0 seeds.*

4. OFF-TARGET EXACT SEEDS

We now turn our attention to the problem of computing the probability that the seeding process is off-target when using exact seeds—recall from section 1.3 that off-target seeding means that the candidate set contains a duplicate but not the target.

If there is no duplicate (i.e., $N = 0$), seeding cannot be off-target, it can only be on-target or null. So from here we assume that the target has $N \geq 1$ duplicates. Let S_0 denote the event that there is an on-target seed and let S_j denote the event that there is a seed for the j-th duplicate. We are thus interested in computing $P\big(\overline{S}_0 \cap (S_1 \cup \ldots \cup S_N)\big)$, where $\overline{S}_j$ denotes the complement of the event S_j. First observe that

$$P\big(\overline{S}_0 \cap (S_1 \cup \ldots \cup S_N)\big) = P(\overline{S}_0) - P\big(\overline{S}_0 \cap \overline{S}_1 \cap \ldots \cap \overline{S}_N\big). \quad (8)$$

Since the duplicates are assumed to evolve independently of each other and through the same mutagenesis process, the events S_j $(1 \leq j \leq N)$ are independent and identically distributed conditionally on $\overline{S}_0$. We can thus write

$$\begin{aligned} P(\overline{S}_0 \cap \ldots \cap \overline{S}_N) &= P(\overline{S}_0) \cdot P(\overline{S}_1 \cap \ldots \cap \overline{S}_N | \overline{S}_0) \\ &= P(\overline{S}_0) \cdot P(\overline{S}_1 | \overline{S}_0)^N = P(\overline{S}_0) \cdot \left(\frac{P(\overline{S}_0 \cap \overline{S}_1)}{P(\overline{S}_0)} \right)^N . (9) \end{aligned}$$

Combining the two equations above, we obtain

$$P\big(\overline{S}_0 \cap (S_1 \cup \ldots \cup S_N)\big) = P(\overline{S}_0) - P(\overline{S}_0) \cdot \left(\frac{P(\overline{S}_0 \cap \overline{S}_1)}{P(\overline{S}_0)} \right)^N . \quad (10)$$

Hence, the probability that seeding is off-target is a function of just two quantities: $P(\overline{S}_0)$ and $P(\overline{S}_0 \cap \overline{S}_1)$. The first is the probability that the read has no seed for the target, which we have already computed in section 3.4 using recursive expression (3). We now need to find a way to compute $P(\overline{S}_0 \cap \overline{S}_1)$.

Remark 2. *Observe that expression (9) is the probability of null seeding (the read contains no seed for the target or any of its duplicates). Since it is also a function of just $P(\overline{S}_0)$ and $P(\overline{S}_0 \cap \overline{S}_1)$, it can be computed at no additional cost. This probability is less relevant than the probabilities that the seeding process is on-target or off-target, but at times, it may be useful to know the probability that a read is not mappable, especially when reads are relatively short.*

4.1. The Dual Encoding

$P(\overline{S}_0 \cap \overline{S}_1)$ is the probability that the read has no seed for the target or for the first duplicate—numbering is arbitrary here, the first duplicate can be any fixed duplicate. As in section 3, we first recode the reads using a specialized alphabet to simplify the problem.

It will be useful to consider a more general problem where we have two sequences of interest labeled $+$ and $-$. The $+$ sequence stands for the target and that the $-$ sequence stands for its duplicate. We then define the dual alphabet $\tilde{\mathcal{A}}_0 = \{\Box, |, \downarrow^{-}_{/1}, \downarrow^{-}_{/2}, \ldots, \downarrow^{+}_{/1}, \downarrow^{+}_{/2}, \ldots, \Downarrow\}$. The symbols $\downarrow^{-}_{/j}$ $(j \geq 1)$ signify that the nucleotide is a mismatch against the $-$ sequence only, the symbols $\downarrow^{+}_{/j}$ $(j \geq 1)$ signify that it is a mismatch against the $+$ sequence only, and the symbol $\Downarrow$ signifies that it is a mismatch against both. As before, every other nucleotide is replaced by the symbol $\Box$, and the terminator $|$ is appended to the end of the read. We again define reads as sequences of segments (zero or more $\Box$ symbols followed by a terminator), except that now the terminators are the symbols $\downarrow^{-}_{/j}$, $\downarrow^{+}_{/j}$ $(j \geq 1)$ and $\Downarrow$. The tail, as usual, is terminated by the symbol $|$.

The index j in the symbol $\downarrow^{-}_{/j}$ indicates the match length of the $+$ sequence (note that this is not the same as the number of $\Box$ symbols in the segment). Likewise, the index j in the symbol $\downarrow^{+}_{/j}$ indicates the match length of the $-$ sequence. For instance, the symbol $\downarrow^{-}_{/7}$ indicates that the nucleotide is a mismatch against the $-$ sequence, that it is a match for the $+$ sequence, and that the six preceding nucleotides were also a match for the $+$ sequence (but the nucleotide before that was a mismatch against the $+$ sequence). The terminators thus encode the local state of the read.

Figure 8 shows an example of read in the dual encoding. The $+$ and $-$ sequences are shown below the read, with matches represented as open squares and mismatches as closed squares. It is visible from this example that symbols $\downarrow^{-}_{/j}$ and $\downarrow^{+}_{/j}$ alternate whenever the mismatches hit different sequences. The symbol $\Downarrow$ occurs only when a nucleotide is a double mismatch.

Let us assume that for each nucleotide, a is the probability that the read matches both sequences, b is the probability that it matches only the $+$ sequence, c is the probability that it matches only the $-$ sequence and d is the probability that it matches neither. Since there are no other cases, we have $a+b+c+d = 1$.

With these definitions, the weighted generating functions of the symbols $\Box$, $\downarrow^{-}_{/j}$, $\downarrow^{+}_{/j}$ $(j \geq 1)$ and $\Downarrow$ are az, bz, cz, and dz, respectively. The next sections clarify how this is used to compute the weighted generating functions of interest.

4.2. Segments Following $\Downarrow$

After a $\Downarrow$ terminator, the match counter for both sequences is reset; the following segment can thus have up to $\gamma - 1$ matches for any of the two sequences. Each match corresponds to the $\Box$ symbol with weighted generating function az. The terminators $\Downarrow$ and $|$ have respective generating function dz and 1 (recall that the tail symbol has size 0), so if the next terminator is $\Downarrow$ or $|$, the segments have weighted generating functions $(1 + az + \ldots + (az)^{\gamma-1})dz$ or $1 + az + \ldots + (az)^{\gamma-1}$, respectively.

If the next terminator is $\downarrow^{-}_{/j}$, there is a match of length j for the $+$ sequence, so the segment contains $j-1$ symbols $\Box$ plus the terminator (which also matches the $+$ sequence). The weighted generating function is thus $(az)^{j-1}bz$. By the same rationale, if the next terminator is $\downarrow^{+}_{/j}$, the weighted generating function of the segment is $(az)^{j-1}cz$.

The terminators $\downarrow^{+}_{/j}$ and $\downarrow^{-}_{/j}$ are disallowed for $j \geq \gamma$ because this would create a seed for at least one of the sequences.

4.3. Segments Following $\downarrow^{+}_{/i}$

At a $\downarrow^{+}_{/i}$ terminator, the match length of the $+$ sequence is 0 and the match length of the $-$ sequence is i. The next segment can thus have up to $\gamma - 1$ matches for the $+$ sequence, but only

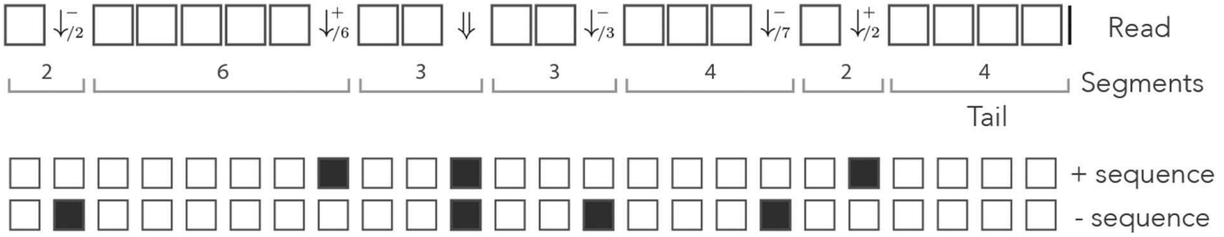


FIGURE 8 | Example of dual encoding. An example of read is represented in the dual alphabet. The symbols $\downarrow^{-}_{/j}$ ($j \geq 1$) represent a mismatch against the $-$ sequence, the symbols $\downarrow^{+}_{/j}$ ($j \geq 1$) represent a mismatch against the $+$ sequence, and the symbol $\Downarrow$ represents a mismatch against both. The index i is the match length of the sequence that is not mismatched. The symbolic $+$ and $-$ sequences are represented below, where an open square stands for a match and a closed square stands for a mismatch.

$\gamma - i - 1$ matches for the $-$ sequence, lowering the maximum size of the segment. If the next terminator is $\Downarrow$ or $|$, the segments have weighted generating functions $(1 + az + \ldots + (az)^{\gamma-j-1})dz$ and $1 + az + \ldots + (az)^{\gamma-j-1}$, respectively.

If the next terminator is $\downarrow^{-}_{/j}$, the weighted generating function is $(az)^{j-1}bz$ as in the previous section. The difference is that the terminators $\downarrow^{-}_{/j}$ are allowed only for $1 \leq j \leq \gamma - i$, otherwise this would create a seed for the $-$ sequence.

If the next terminator is $\downarrow^{+}_{/j}$, the situation is slightly more complex because this imposes $i < j \leq \gamma - 1$. Indeed, there were already i matches for the $-$ sequence at the terminator $\downarrow^{+}_{/i}$, and there will be more at the end of the following segment because it has no mismatch for the $-$ sequence. Taking this into account, we see that the weighted generating function of those segments is $(az)^{j-i-1}cz$ with $i < j \leq \gamma - 1$.

4.5. Transfer Matrix

We now have all the elements to specify the transfer matrix of reads with no seed for either sequence. Recall that a is the probability of a double match, b is the probability of a mismatch only against the $-$ sequence, c is the probability of a mismatch only against the $+$ sequence and d is the probability of a double mismatch. For notational convenience, we define

$$\begin{aligned} r_i^{+}(z) &= (az)^i cz, \\ r_i^{-}(z) &= (az)^i bz, \\ R_i(z) &= \big(1 + az + \ldots + (az)^i\big)dz, \\ F_i(z) &= 1 + az + \ldots + (az)^i. \end{aligned} \tag{11}$$

With these notations, the information from the previous sections can be summarized in the transfer matrix $\tilde{M}_0(z)$ equal to

$$\begin{array}{c|cccccccc|} & \Downarrow & \downarrow^{+}_{/1} & \cdots & \downarrow^{+}_{\gamma-1} & \downarrow^{-}_{/1} & \cdots & \downarrow^{-}_{/\gamma-1} & | \\ \hline \Downarrow & R_{\gamma-1}(z) & r_0^{+}(z) & \cdots & r_{\gamma-2}^{+}(z) & r_0^{-}(z) & \cdots & r_{\gamma-2}^{-}(z) & F_{\gamma-1}(z) \\ \downarrow^{+}_{/1} & R_{\gamma-2}(z) & & & & & & & F_{\gamma-2}(z) \\ \downarrow^{+}_{/2} & R_{\gamma-3}(z) & & & & & & & F_{\gamma-3}(z) \\ \vdots & \vdots & & \tilde{A}(z) & & & \tilde{B}_0(z) & & \vdots \\ \downarrow^{+}_{/\gamma-1} & R_0(z) & & & & & & & F_0(z) \\ \downarrow^{-}_{/1} & R_{\gamma-2}(z) & & & & & & & F_{\gamma-1}(z) \\ \downarrow^{+}_{/2} & R_{\gamma-3}(z) & & & & & & & F_{\gamma-2}(z) \\ \vdots & \vdots & & \tilde{C}_0(z) & & & \tilde{D}(z) & & \vdots \\ \downarrow^{+}_{/\gamma-1} & R_0(z) & & & & & & & F_0(z) \\ | & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \hline \end{array},$$

4.4. Segments Following $\downarrow^{-}_{/i}$

We can find the weighted generating functions by just reversing the $+$ and $-$ signs in the previous section. This way, we see that the weighted generating function of the segments terminated by $\Downarrow$ or $|$ are $(1 + az + \ldots + (az)^{\gamma-i-1})dz$ and $1 + az + \ldots + (az)^{\gamma-i-1}$, respectively.

Likewise, the weighted generating function of the segments terminated by $\downarrow^{+}_{/j}$ is $(az)^{j-1}cz$, where $1 \leq j \leq \gamma - i$; and the weighted generating function of the segments terminated by $\downarrow^{-}_{/j}$ is $(az)^{j-i-1}bz$ where $i < j \leq \gamma - 1$.

where γ is the minimum seed length, and where $\tilde{A}(z)$, $\tilde{B}_0(z)$, $\tilde{C}_0(z)$, and $\tilde{D}(z)$ are matrices of dimensions $(\gamma - 1) \times (\gamma - 1)$ that are defined as

$$\tilde{A}(z) = \begin{array}{c|ccccc|} & \downarrow^{+}_{/1} & \downarrow^{+}_{/2} & \cdots & \downarrow^{+}_{/\gamma-2} & \downarrow^{+}_{/\gamma-1} \\ \hline \downarrow^{+}_{/1} & 0 & r_0^{+}(z) & \cdots & r_{\gamma-4}^{+}(z) & r_{\gamma-3}^{+}(z) \\ \downarrow^{+}_{/2} & 0 & 0 & \cdots & r_{\gamma-5}^{+}(z) & r_{\gamma-4}^{+}(z) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \downarrow^{+}_{/\gamma-2} & 0 & 0 & \cdots & 0 & r_0^{+}(z) \\ \downarrow^{+}_{/\gamma-1} & 0 & 0 & \cdots & 0 & 0 \\ \hline \end{array},$$

$$\tilde{B}_0(z) = \begin{array}{cc} & \begin{array}{ccccc} \downarrow^{-}_{/1} & \downarrow^{-}_{/2} & \cdots & \downarrow^{-}_{/\gamma-2} & \downarrow^{-}_{/\gamma-1} \end{array} \\ \begin{array}{c} \downarrow^{+}_{/1} \\ \downarrow^{+}_{/2} \\ \vdots \\ \downarrow^{+}_{/\gamma-2} \\ \downarrow^{+}_{/\gamma-1} \end{array} & \left[\begin{array}{ccccc} r^{-}_{0}(z) & r^{-}_{1}(z) & \dots & r^{-}_{\gamma-3}(z) & r^{-}_{\gamma-2}(z) \\ r^{-}_{0}(z) & r^{-}_{1}(z) & \dots & r^{-}_{\gamma-3}(z) & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r^{-}_{0}(z) & r^{-}_{1}(z) & \dots & 0 & 0 \\ r^{-}_{0}(z) & 0 & \dots & 0 & 0 \end{array}\right] \end{array},$$

$$\tilde{C}_0(z) = \begin{array}{cc} & \begin{array}{ccccc} \downarrow^{+}_{/1} & \downarrow^{+}_{/2} & \cdots & \downarrow^{+}_{/\gamma-2} & \downarrow^{+}_{/\gamma-1} \end{array} \\ \begin{array}{c} \downarrow^{-}_{/1} \\ \downarrow^{-}_{/2} \\ \vdots \\ \downarrow^{-}_{/\gamma-2} \\ \downarrow^{-}_{/\gamma-1} \end{array} & \left[\begin{array}{ccccc} r^{+}_{0}(z) & r^{+}_{1}(z) & \dots & r^{+}_{\gamma-3}(z) & r^{+}_{\gamma-2}(z) \\ r^{+}_{0}(z) & r^{+}_{1}(z) & \dots & r^{+}_{\gamma-3}(z) & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r^{+}_{0}(z) & r^{+}_{1}(z) & \dots & 0 & 0 \\ r^{+}_{0}(z) & 0 & \dots & 0 & 0 \end{array}\right] \end{array},$$

$$\tilde{D}(z) = \begin{array}{cc} & \begin{array}{ccccc} \downarrow^{-}_{/1} & \downarrow^{-}_{/2} & \cdots & \downarrow^{-}_{/\gamma-2} & \downarrow^{-}_{/\gamma-1} \end{array} \\ \begin{array}{c} \downarrow^{-}_{/1} \\ \downarrow^{-}_{/2} \\ \vdots \\ \downarrow^{-}_{/\gamma-2} \\ \downarrow^{-}_{/\gamma-1} \end{array} & \left[\begin{array}{ccccc} 0 & r^{-}_{0}(z) & \dots & r^{-}_{\gamma-4}(z) & r^{-}_{\gamma-3}(z) \\ 0 & 0 & \dots & r^{-}_{\gamma-5}(z) & r^{-}_{\gamma-4}(z) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & r^{-}_{0}(z) \\ 0 & 0 & \dots & 0 & 0 \end{array}\right] \end{array}.$$

As before, the term of interest is the top right entry of $\tilde{M}_0(z) \cdot (I - \tilde{M}_0(z))^{-1}$. To see why, observe that every read can be prepended by $\Downarrow$-segments and only by those (every other terminator would imply that one of the two sequences has a nonzero match size at the start of the read). Thus, reads are precisely the sequences of segments that can follow the symbol $\Downarrow$ and that are terminated by a tail, the weighted generating function of which is the top right entry of the matrix.

$\tilde{M}_0(z)$ is too complex to compute a closed expression of $\tilde{M}_0(z) \cdot (I - \tilde{M}_0(z))^{-1}$. It is easier to proceed as in section 3.6 and to compute the powers of $\tilde{M}_0(z)$ up to a finite value. This is done once again using the arithmetic of truncated polynomials. Since each segment except the tail contains a mismatch against at least one sequence, the top right entry of $\tilde{M}_0(z)^{s+1}$ is the weighted generating function of reads that contain s mismatches (where double mismatches count as one). We thus define $\tilde{p}$ as the upper bound on the probability of a mismatch, *i.e.* $\tilde{p} = \max\{b, c, d\}$. The updated formula (7) now gives an upper bound of the probability that a read of size k contains s or more mismatches as

$$Pr(X \geq s) \leq \exp\left((s-k)\log\frac{k-s}{k(1-\tilde{p})} - s\log\frac{s}{k\tilde{p}}\right).$$

With this upper bound, we can compute the terms of the matrix partial sums $\tilde{M}_0(z) + \tilde{M}_0(z)^2 + \ldots + \tilde{M}_0(z)^s$ until the ignored terms become negligible, *i.e.* until we can be sure that the coefficient of interest a_k is accurate to within chosen ε.

Now returning to the problem of computing $P(\overline{S}_0 \cap \overline{S}_1)$, the $+$ sequence is interpreted as the target and the $-$ sequence as the duplicate. Based on the assumptions of the error model presented in section 3.1, this implies that $a = (1-p)(1-\mu)$, $b = (1-p)\mu$, $c = p\mu/3$, and $d = p(1-\mu/3)$.

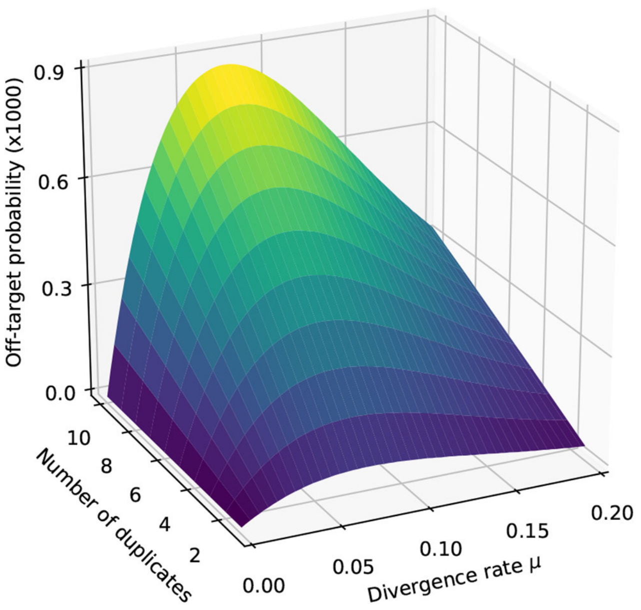


FIGURE 9 | Off-target seeding probabilities (exact seeds). The surfaces show the probability that seeding is off-target as a function of the divergence rate μ and of the number N of duplicates. The results are shown for exact seeds of size $\gamma = 19$ and for reads of size $k = 50$ nucleotides sequenced with an error rate $p = 0.01$. The divergence rate μ is defined as the probability that a given duplicate differs from the target at any given position. Note the factor 1,000 in the scale of the z-axis.

With these values, we can fully specify the matrix $\tilde{M}_0(z)$, and compute its powers until the estimate of the coefficient of interest a_k is accurate enough, finally giving a numerical value for $P(\overline{S}_0 \cap \overline{S}_1)$.

4.6. Illustration

We illustrate the strategy delineated above for reads of size $k = 50$ sequenced with an instrument with error rate $p = 0.01$, when using exact seeds of size $\gamma = 19$.

Figure 9 shows the result for a number of duplicates N from 1 to 10 and for a divergence rate μ from 0 to 0.20. The first observation is that the probability that seeding is off-target increases with N. This is also clear from expression (10). This can also be understood intuitively because the probability of not seeding the target is fixed and the probability of having an empty candidate set decreases as N increases. As a result the probability that the candidate set contains only invalid candidates increases with N.

The second observation is that there exists a "worst" value of μ situated around 0.070. When μ is much smaller, the duplicates tend to be exactly identical to the target, making it impossible that there is a seed for a duplicate but no seed for the target. When μ is much larger, the duplicates are far from the target and they are unlikely to be in the candidate set at all. In expression (10), the only term that depends on μ is $P(\overline{S}_0 \cap \overline{S}_1)$, and it is clear that the minimum of expression (10) corresponds to the maximum of $P(\overline{S}_0 \cap \overline{S}_1)$. This is why the worst value of μ is the same for every N.

5. OFF-TARGET SKIP SEEDS

To compute the probability that the seeding process is off-target when using skip seeds, we observe that the logic of section 4 can be transposed with few modifications. In particular, the probability can be computed through expression (10), where S_0 is the event that the read has a *skip* seed for the target (instead of an exact seed) and S_1 is the event that the read has a *skip* seed for the first duplicate (instead of an exact seed).

We have already seen how to compute $P(\overline{S}_0)$ in section 3.6, we now need to find a way to compute $P(\overline{S}_0 \cap \overline{S}_1)$ when using skip seeds.

5.1. The Skip Dual Encoding

As before, we recode the reads in a specialized alphabet. We define the skip-n dual alphabet as $\tilde{\mathcal{A}}_n = \{\square, *, |, \Downarrow_0, \Downarrow_1, \Downarrow_2, \ldots, \Downarrow_n, \downarrow^{-}_{/1}, \downarrow^{-}_{/2}, \ldots, \downarrow^{+}_{/1}, \downarrow^{+}_{/2}, \ldots\}$. The symbols $\square$, $|$, $\downarrow^{-}_{/j}$ and $\downarrow^{+}_{/j}$ $(j \geq 1)$ have the same meaning as in the dual alphabet of section 4.1, i.e., the $\square$ symbol stands for a double match, the $|$ terminator marks the end of the read, the $\downarrow^{-}_{/j}$ symbol indicates a mismatch against the $-$ sequence with a match length of size j for the $+$ sequence, and conversely the $\downarrow^{+}_{/j}$ symbol indicates a mismatch against the $+$ sequence with a match of length j for the $-$ sequence. The symbols $\Downarrow_j$ $(0 \leq j \leq n)$ indicate that both sequences have match length 0 and that the next non-skipped position is j nucleotides further. The symbol $*$ indicates that it does not matter whether the nucleotide is a match or a mismatch, as we will explain below.

Figure 10 shows the read from **Figure 8** represented in the skip-3 dual encoding. It is important to note several differences with **Figure 8**. The first is that the symbols $\Downarrow_j$ $(0 \leq j \leq n)$ are not always associated with double mismatches. For instance, the symbol $\Downarrow_0$ on the right side of the read corresponds to a mismatch for the $+$ sequence only. This happens whenever the $+$ and the $-$ sequences are mismatched in the same interval between non-skipped positions (the mismatches do not need to be on the same nucleotide).

Also observe that the symbol $\Downarrow_1$ is followed by a $*$ symbol, indicating that it does not matter whether the nucleotide is a match or a mismatch for any of the two sequences. After the $\Downarrow_1$ symbol, both sequences have match length 0 and we have to "wait" for a potential new seed one nucleotide further downstream. As in **Appendix B.3**, the $\Downarrow_j$ symbols are followed by j symbols $*$, unless the end of the read comes before the next non-skipped position. After a sequence of $*$ symbols, the read is either finished or at a non-skipped position. In the second case, the is in the same state as after a $\Downarrow_0$ symbol, so we can consider that the $*$ symbol allows the terminators $\Downarrow_j$ $(1 \leq j \leq n)$ to just "fast forward" to either $\Downarrow_0$ or $|$, as shown in **Appendix B.3**.

As in the previous section, let a, b, c, and d denote the probabilities of a double match, a mismatch against $-$, a mismatch against $+$ and a double mismatch, respectively. With these definitions, the weighted generating functions of the symbols $\square$, $\downarrow^{-}_{/j}$ and $\downarrow^{+}_{/j}$ $(j \geq 1)$ are az, bz, and cz, respectively. The weighted generating function of the $*$ symbol is z, and that of the symbols $\Downarrow_j$ $(0 \leq j \leq n)$ will be worked out in the sections below.

5.2. Segments Following $\Downarrow_i$ $(1 \leq i \leq n)$

A $\Downarrow_i$ terminator is followed by up to i symbols $*$. If there are fewer than i symbols $*$, the read is finished so the segment must be a tail. Recall that the weighted generating function of the $|$ terminator is 1, so the weighted generating function of tail segments is $1 + z + \ldots + z^{i-1}$.

If there are i symbols $*$, the sequence ends at a non-skipped position, *i.e.* in the same state as after a $\Downarrow_0$ terminator. This is not a segment proper, because there is no terminator, but in the transfer matrix the symbols $\Downarrow_i$ $(1 \leq i \leq n)$ project directly to the symbol $\Downarrow_0$ with weighted generating function z^i.

5.3. Segments Following $\Downarrow_0$

After a $\Downarrow_0$ symbol, all the counters are reset as in the beginning of the read. If the next terminator is $|$, the segment is a tail and we only have to make sure that it contains fewer than γ symbols $\square$, otherwise this would create a seed. The weighted generating function of tail segments following $\Downarrow_0$ is thus $1 + qz + \ldots + (qz)^{\gamma-1}$.

If the next terminator is a $\downarrow^{-}_{/j}$ symbol $(j \geq 1)$, the segment is a match of size j for the $+$ sequence. This imposes $j < \gamma$ otherwise the segment would create a seed. Such segments consist of $j - 1$ symbols $\square$ followed by the terminator, so their weighted generating function is $(az)^{j-1}bz$ with $1 \leq j < \gamma$.

Conversely, if the next terminator is a $\downarrow^{+}_{/j}$ symbol $(j \geq 1)$, we can apply the same rationale to see that the weighted generating function is $(az)^{j-1}cz$, where $1 \leq j < \gamma$.

Finally, if the next terminator is a $\Downarrow_j$ symbol $(0 \leq j \leq n)$, we must not only make sure that the segment contains fewer than γ symbols $\square$, but also keep track of the position of the next non-skipped position (stored in index j). Here we can follow verbatim the rationale of section 3.6 where we replace the probability of a match (previously q) by that of a double match (now a), and the probability of a mismatch (previously p) by that a double mismatch (now d). Replacing the symbols in expression (4), we see that the weighted generating function is $(az)^x \cdot \left(1 + (az)^{n+1} + \ldots + (az)^{(n+1)m}\right) dz$, where $x = n - j \pmod{n+1}$ and $m = \lfloor (\gamma - 1 - x)/(n+1) \rfloor$.

5.4. Segments Following $\downarrow^{-}_{/i}$

At a $\downarrow^{-}_{/i}$ terminator $(i \geq 1)$, the read contains a match of length i for the $+$ sequence. If the next segment is a tail, we must make sure that the match length for the $+$ sequence does not exceed $\gamma - 1$. This means that we can have up to $\gamma - i - 1$ symbols $\square$ and thus that the weighted generating function of the tail segments is $1 + (qz) + \ldots + (qz)^{\gamma-i-1}$.

If the next terminator is a $\downarrow^{-}_{/j}$ symbol, we must have $j > i$ because there is no mismatch against the $+$ sequence. In this case, we must only make sure that the total match length for the $+$ sequence remains lower than γ. Such segments contain $j - i - 1$ symbols $\square$ followed by the terminator so their weighted generating function is $(qz)^{i-j-1}bz$ with $i < j \leq \gamma - 1$.

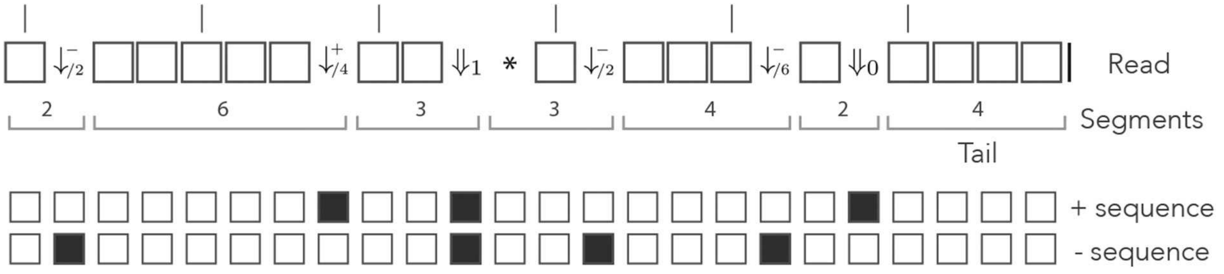


FIGURE 10 | Example of skip dual encoding. The read of **Figure 8** is represented in the skip dual alphabet. The vertical bars above the read indicate non-skipped positions. The symbols $\downarrow_{/j}^{-}$ and $\downarrow_{/j}^{+}$ ($j \geq 1$) have the same meaning as in the dual alphabet. The symbols $\Downarrow_j$ indicate that both sequences have match length 0 and that the next non-skipped position is located j nucleotides downstream. Other features are as in **Figure 8**.

For the last two types of terminators, we must pay attention to the fact that in general, a mismatch against + that follows a mismatch against − can be represented by either a $\Downarrow_j$ symbol ($0 \leq j \leq n$) or a $\downarrow_{/j}^{+}$ symbol ($j \geq 1$). The terminator is $\Downarrow_j$ if the two mismatches are within the same interval between non-skipped positions because both sequences locally have match length 0. If the mismatch against + is in another interval, the terminator is $\downarrow_{/j}^{+}$ because in that case the − sequence has a positive match length. For now, bear in mind that a mismatch against the + sequence only can produce a $\Downarrow_j$ terminator ($0 \leq j \leq n$), this will be important later.

If the next terminator is $\downarrow_{/j}^{+}$ ($j \geq 1$), then it must be separated from the preceding $\downarrow_{/i}^{-}$ terminator by a non-skipped position, which imposes a lower bound on the size of the segment. Since at the $\downarrow_{/i}^{-}$ terminator the match length for + was i, there must be a non-skipped position i nucleotides before. The number of nucleotides from the $\downarrow_{/i}^{-}$ terminator to the next non-skipped position is thus $y = -i \pmod{n+1}$, so the minimum segment length is $y + 1$. There is also an upper bound on the size of the segment because the match length for the + sequence cannot become higher than $\gamma - 1$, imposing the length to be lower than $\gamma - i$. The shortest segment is terminated by $\downarrow_{/1}^{+}$, and the longest by $\downarrow_{/\gamma-y-i}^{+}$. Finally, the weighted generating function of segments terminated by $\downarrow_{/j}^{+}$ following the $\downarrow_{/i}^{-}$ terminator is $(az)^{y+j-1}cz$ with $1 \leq j \leq \gamma - y - i$. Note that for some value of i and y, no j can satisfy the last inequality.

Finally, if the next terminator is $\Downarrow_j$ ($0 \leq j \leq n$) we have to distinguish two cases, depending on whether the segment ends with a double mismatch or with a single mismatch against the + sequence. For the first case we can apply the rationale of section 3.6. Recall that the index j indicates the number of nucleotides until the next non-skipped position. Let ℓ_0 be the length of the shortest possible segment. The lengths of the other segments are of the form $\ell_0 + n + 1, \ldots, \ell_0 + m(n+1)$, where the integer m must be chosen so that the segment does not create a seed. At the $\downarrow_{/i}^{-}$ terminator, there is a match of length i for the + sequence so m is the largest integer such that $i + \ell_0 + m(n+1) < \gamma$, *i.e.* $m = \lfloor (\gamma - i - \ell_0)/(n+1) \rfloor$.

The $\downarrow_{/i}^{-}$ terminator is $y = -i \pmod{n+1}$ nucleotides before the next non-skipped position so the shortest segment has length $\ell_0 = y - j$ if $y > j$, and $n + 1 - j + y$ otherwise. This is equivalent to defining ℓ_0 as $x + 1$ where x is the number of $\square$ symbols in the shortest segment, i.e., $x = -i - j - 1 \pmod{n+1}$. Summing the weighted generating functions of the individual segments, we find $(az)^x(1 + (az)^{n+1} + \ldots + (az)^{m(n+1)})dz$.

The last remaining issue is that a mismatch against the + sequence only can produce a $\Downarrow_j$ terminator. This happens when the preceding $\downarrow_{/i}^{-}$ terminator is not separated from the mismatch by a non-skipped position (in this case both sequences locally have match length 0). The $\downarrow_{/i}^{-}$ terminator is located y nucleotides before the next non-skipped position, so if the terminator is from $\Downarrow_0$ to $\Downarrow_{y-1}$ we need to add the term $(az)^x bz$ to the previous weighted generating function. In conclusion, the weighted generating function of segments terminated by $\Downarrow_j$ is $(az)^x(1 + (az)^{n+1} + \ldots + (az)^{m(n+1)})dz + \delta_{i,j}^{+}(z)$, where $\delta_{i,j}^{+}(z) = (az)^x bz$ if $j < y$ and 0 otherwise.

5.5. Segments Following $\downarrow_{/i}^{+}$

We can find the weighted generating functions by just reversing the + and − signs in the previous section. This way we can see that the weighted generating function of tail segments is $1 + (qz) + \ldots + (qz)^{\gamma-i-1}$.

Likewise, the weighted generating function of segments terminated by $\downarrow_{/j}^{+}$ is $(qz)^{i-j-1}bz$ with $i < j \leq \gamma - 1$.

The weighted generating function of segments terminated by $\downarrow_{/j}^{-}$ is $(az)^{y+j-1}dz$ with $y = -i \pmod{n+1}$ and $1 \leq j \leq \gamma - y - i$.

Finally, the weighted generating function of segments terminated by $\Downarrow_j$ is $(az)^x(1 + (az)^{n+1} + \ldots + (az)^{m(n+1)})dz + \delta_{i,j}^{-}(z)$, where $\delta_{i,j}^{-}(z) = (az)^x cz$ if $j < y$ and 0 otherwise.

5.6. Transfer Matrix

We now have all the elements to specify the transfer matrix of reads with no skip-n seed for either sequence. Recall that n is the number of skipped nucleotides, γ is the minimum seed length, a is the probability of a double match, b is the probability of a mismatch against the − sequence only, c is the probability of a mismatch against the + sequence only and d is the probability of a double mismatch. For notational convenience we define

$$N_i(z) = 1 + z + \ldots + z^i, \tag{12}$$

$$W_j(z) = (az)^x\big(1 + (az)^{n+1} + \ldots + (az)^{(n+1)m}\big)dz, \tag{13}$$

$$\text{where } x = -j - 1 \pmod{n+1}, \text{ and } m = \left\lfloor \frac{\gamma - 1 - x}{n+1} \right\rfloor,$$

$$U_{i,j}(z) = (az)^x\big(1 + (az)^{n+1} + \ldots + (az)^{(n+1)m}\big)dz + \begin{cases} bz \cdot (az)^x \text{ if } j < y, \\ 0 \text{ otherwise} \end{cases} \tag{14}$$

$$\text{where } x = -i - j - 1 \pmod{n+1}, y = -i \pmod{n+1},$$

$$\text{and } m = \left\lfloor \frac{\gamma - 1 - i - x}{n+1} \right\rfloor, \quad (14)$$

$$V_{i,j}(z) = (az)^x\big(1 + (az)^{n+1} + \ldots + (az)^{(n+1)m}\big)dz + \begin{cases} cz \cdot (az)^x \text{ if } j \le x, \\ 0 \text{ otherwise} \end{cases} \tag{15}$$

$$\text{where } x = -i - j - 1 \pmod{n+1}, m = \left\lfloor \frac{\gamma - 1 - i - x}{n+1} \right\rfloor.$$

With these notations, the information from the previous sections can be summarized in the transfer matrix $\tilde{M}_n(z)$ equal to

$$\begin{array}{c|ccccccccccc|}
 & \Downarrow_0 & \Downarrow_1 & \cdots & \Downarrow_n & \downarrow^{+}_{/1} & \cdots & \downarrow^{+}_{\gamma-1} & \downarrow^{-}_{/1} & \cdots & \downarrow^{-}_{/\gamma-1} & | \\
\Downarrow_0 & W_0(z) & W_1(z) & \ldots & W_n(z) & r^{+}_0(z) & \ldots & r^{+}_{\gamma-2}(z) & r^{-}_0(z) & \ldots & r^{-}_{\gamma-2}(z) & F_{\gamma-1}(z) \\
\Downarrow_1 & z & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & N_0(z) \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\Downarrow_n & z^n & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & N_{n-1}(z) \\
\downarrow^{+}_{/1} & U_{1,0}(z) & U_{1,1}(z) & \ldots & U_{1,n}(z) & & & & & & & F_{\gamma-2}(z) \\
\vdots & \vdots & \vdots & \ddots & \vdots & & \tilde{A}(z) & & & \tilde{B}_n(z) & & \vdots \\
\downarrow^{+}_{/\gamma-1} & U_{\gamma-1,0}(z) & U_{\gamma-1,1}(z) & \ldots & U_{\gamma-1,n}(z) & & & & & & & F_0(z) \\
\downarrow^{-}_{/1} & V_{1,0}(z) & V_{1,1}(z) & \ldots & V_{1,n}(z) & & & & & & & F_{\gamma-2}(z) \\
\vdots & \vdots & \vdots & \ddots & \vdots & & \tilde{C}_n(z) & & & \tilde{D}(z) & & \vdots \\
\downarrow^{-}_{/\gamma-1} & V_{\gamma-1,0}(z) & V_{\gamma-1,1}(z) & \ldots & V_{\gamma-1,n}(z) & & & & & & & F_0(z) \\
| & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0
\end{array}.$$

The matrices $\tilde{A}(z)$ and $\tilde{C}(z)$ in the expression of $\tilde{M}_n(z)$ are the same as in section 4.1. They are reproduced here for convenience.

$$\tilde{A}(z) = \begin{array}{c|ccccc|}
 & \downarrow^{+}_{/1} & \downarrow^{+}_{/2} & \cdots & \downarrow^{+}_{/\gamma-2} & \downarrow^{+}_{/\gamma-1} \\
\downarrow^{+}_{/1} & 0 & r^{+}_0(z) & \ldots & r^{+}_{\gamma-4}(z) & r^{+}_{\gamma-3}(z) \\
\downarrow^{+}_{/2} & 0 & 0 & \ldots & r^{+}_{\gamma-5}(z) & r^{+}_{\gamma-4}(z) \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\downarrow^{+}_{/\gamma-2} & 0 & 0 & \ldots & 0 & r^{+}_0(z) \\
\downarrow^{+}_{/\gamma-1} & 0 & 0 & \ldots & 0 & 0
\end{array},$$

$$\tilde{C}_0(z) = \begin{array}{c|ccccc|}
 & \downarrow^{+}_{/1} & \downarrow^{+}_{/2} & \cdots & \downarrow^{+}_{/\gamma-2} & \downarrow^{+}_{/\gamma-1} \\
\downarrow^{-}_{/1} & r^{+}_0(z) & r^{+}_1(z) & \ldots & r^{+}_{\gamma-3}(z) & r^{+}_{\gamma-2}(z) \\
\downarrow^{-}_{/2} & r^{+}_0(z) & r^{+}_1(z) & \ldots & r^{+}_{\gamma-3}(z) & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\downarrow^{-}_{/\gamma-2} & r^{+}_0(z) & r^{+}_1(z) & \ldots & 0 & 0 \\
\downarrow^{-}_{/\gamma-1} & r^{+}_0(z) & 0 & \ldots & 0 & 0
\end{array}.$$

The matrices $\tilde{B}_n(z)$ and $\tilde{C}_n(z)$ are defined as

$$\tilde{B}_n(z) = \begin{array}{c|ccccc|}
 & \downarrow^{-}_{/1} & \downarrow^{-}_{/2} & \cdots & \downarrow^{-}_{/\gamma-2} & \downarrow^{-}_{/\gamma-1} \\
\downarrow^{+}_{/1} & s_{1,1}(z) & s_{1,2}(z) & \ldots & s_{1,\gamma-2}(z) & s_{1,\gamma-1}(z) \\
\downarrow^{+}_{/2} & s_{2,1}(z) & s_{2,2}(z) & \ldots & s_{2,\gamma-2}(z) & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\downarrow^{+}_{/\gamma-2} & s_{\gamma-2,1}(z) & s_{\gamma-2,2}(z) & \ldots & 0 & 0 \\
\downarrow^{+}_{/\gamma-1} & s_{\gamma-1,1}(z) & 0 & \ldots & 0 & 0
\end{array},$$

$$\tilde{C}_n(z) = \begin{array}{c|ccccc|}
 & \downarrow^{+}_{/1} & \downarrow^{+}_{/2} & \cdots & \downarrow^{+}_{/\gamma-2} & \downarrow^{+}_{/\gamma-1} \\
\downarrow^{-}_{/1} & t_{1,1}(z) & t_{1,2}(z) & \ldots & t_{1,\gamma-2}(z) & t_{1,\gamma-1}(z) \\
\downarrow^{-}_{/2} & t_{2,1}(z) & t_{2,2}(z) & \ldots & t_{2,\gamma-2}(z) & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\downarrow^{-}_{/\gamma-2} & t_{\gamma-2,1}(z) & t_{\gamma-2,2}(z) & \ldots & 0 & 0 \\
\downarrow^{-}_{/\gamma-1} & t_{\gamma-1,1}(z) & 0 & \ldots & 0 & 0
\end{array},$$

with

$$s_{i,j} = \begin{cases} cz \cdot (az)^{y+j-1} & \text{if } i + j + y \le \gamma \\ 0 & \text{otherwise,} \end{cases}$$

$$t_{i,j} = \begin{cases} bz \cdot (az)^{y+j-1} & \text{if } i + j + y \le \gamma \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{where } y = -i \pmod{n+1}, \text{ in both cases.}$$

Now returning to the problem of computing $P(\overline{S}_0 \cap \overline{S}_1)$, the + sequence is interpreted as the target and the − sequence as the duplicate. Based on the assumptions of the error model presented in section 3.1, this implies that $a = (1-p)(1-\mu)$, $b = (1-p)\mu$, $c = p\mu/3$, and $d = p(1-\mu/3)$.

The computation is performed as described in section 4.1. We compute the successive powers of $\tilde{M}_n(z)$ in the arithmetic of truncated polynomials and stop the iterations using the same criterion. The only modification is that the top right term of $\tilde{M}_n(z)^{s+1}$ is not the weighted generating function of reads with s mismatches (where double mismatches count as one). The reason is that the sequences of $*$ symbols are not terminated by a mismatch. However, there can be at most one sequence of $*$ symbols for each mismatch or double mismatch, so the reads described by the top right term of $\tilde{M}_n(z)^{s+1}$ have at least $\lfloor s/2 \rfloor$ mismatches.

We thus define $\tilde{p}$ as the upper bound on the probability of a mismatch, i.e., $\tilde{p} = \max\{b, c, d\}$ and use the updated formula (7) from section 4.5

$$Pr(X \geq s) \leq \exp\left((s-k) \log \frac{k-s}{k(1-\tilde{p})} - s \log \frac{s}{k\tilde{p}} \right).$$

With this upper bound, we can compute the terms of the partial sums $\tilde{M}_0(z) + \tilde{M}_0(z)^2 + \ldots + \tilde{M}_0(z)^{2s+1}$ until the ignored terms become negligible, *i.e.* until we can be sure that the coefficient of interest a_k is accurate to within chosen ε.

Remark 3. *Observe that when $n = 0$ the matrix $\tilde{M}_n(z)$ is identical to the matrix $\tilde{M}_0(z)$ of section 4.1, again consistent with the fact that exact seeds are skip-0 seeds. The same applies to $\tilde{B}_n(z)$ and $\tilde{C}_n(z)$.*

5.7. Illustration

We illustrate the strategy delineated above using the same settings as in section 4.6 (reads of length $k = 50$, probability of sequencing error $p = 0.01$ and seeds of minimum size $\gamma = 19$), except that we replace exact seeds by skip-5 and skip-9 seeds.

Figure 11 shows the result for a number of duplicates N from 1 to 10 and for a divergence rate μ from 0 to 0.20. The surfaces have the same general aspect as in **Figure 9**. The probability that seeding is off-target increases with N and there is again a worst value of μ, because the maximum of $P(\overline{S}_0 \cap \overline{S}_1)$ minimizes expression (10) for every value of N. However, those values of μ are not the same (they are approximately 0.065 and 0.060 for skip-5 and skip-9 seeds, respectively).

Importantly, **Figure 11** reveals that skipping 5 nucleotides increases the chances that seeding is off-target by a factor approximately 1.5, and skipping 9 nucleotides by a factor approximately 1.8 (compare with **Figure 9**). Skipping nucleotides seems to have little effect, but this is not a general conclusion. Indeed, skipping nucleotides decreases the probability of on-target seeding (skipping positions implies fewer on-target seeds) but increases the probability of null seeding (skipping positions implies fewer seeds overall), so the effects must be evaluated on a case-by-case basis. Skipping nucleotides can even decrease the probability that seeding is off-target. Concretely, for exact seeds of size $\gamma = 19$ in reads of size $k = 50$ with $N = 10$ duplicates at divergence $\mu = 0.1$ with a sequencing error $p = 0.1$, the probability of off-target seeding is approximately 0.178 with exact seeds and 0.035 with skip-9 seeds, showing that skipping nucleotides can have different effects.

This kind of information is critical for choosing the best seeding strategy. Yet, the off-target seeding probability is not the only criterion. Equally important considerations are the probability of on-target seeding, the computational resources required to implement a particular seeding strategy and other sources of mapping errors (see discussion in section 8.1). The benefit of a theory to compute seeding probabilities is to have access to this knowledge.

6. OFF-TARGET MEM SEEDS

MEM seeds are substantially more complex than exact seeds and skip seeds because we need to take into account all the duplicates in the combinatorial construction.

6.1. Hard and Soft Masking

We first introduce two important notions that will be the key to understanding the behavior of MEM seeds.

Definition 8. *At a given position of the read, a duplicate is a hard mask if its match length on the left side is strictly longer than the match length of the target. A duplicate is a soft mask if it has the same match length as the target.*

Figure 12 gives a graphical intuition of hard and soft masks. It is important to bear in mind that hard and soft masks depend on the position of interest: a sequence can be a mask at the left end of the read and not at the right end, or the opposite.

Hard and soft masks explain the counter-intuitive properties of MEM seeds. For instance, in **Figure 4** the target cannot be discovered because every nucleotide of the read has a hard mask. In **Figure 5**, the target could be discovered if the read were shorter because a hard mask would turn into a soft one.

From the definition, we see that the last nucleotide of every strict on-target MEM seed is always unmasked. Conversely, an unmasked nucleotide always belongs to exactly one strict on-target MEM (not necessarily a seed because the size of the MEM can be less than γ). Also, the last nucleotide of every shared on-target MEM seed is always soft-masked, but a soft-masked nucleotide does not always belong to a shared on-target MEM.

Since hard and soft masks inform us about the positions of on-target MEM seeds, we construct an alphabet that encodes the masking status of the nucleotides.

6.2. The MEM Alphabet

As before, we recode the reads as sequences of letters from a specialized alphabet called the MEM alphabet $\mathcal{A} = \{\Box, |, \uparrow_{/1}, \uparrow_{/2}, \uparrow_{/3} \ldots, \downarrow_{/0}, \downarrow_{/1}, \downarrow_{/2}, \ldots\}$.

The symbols $\downarrow_{/m}$ ($m \geq 0$) indicate that the nucleotide is a sequencing error and m is the number of duplicates that *match* the nucleotide. Since a sequencing error is always a mismatch against the target, the symbol $\downarrow_{/0}$ indicates that the nucleotide is a mismatch against *every* sequence. The symbols $\uparrow_{/i}$ indicate a change in masking status: the nucleotide is not masked but the previous is—this happens when all the masks fail to extend beyond this position. The index $i \geq 1$ is the number of nucleotides since the last mismatch or since the beginning of the read. All the other nucleotides are represented by the symbol $\Box$, implying that $\Box$ symbols are never sequencing errors and always match the target. The symbol | is appended to the end of the read as before.

Note that in the symbols $\downarrow_{/m}$ and $\uparrow_{/i}$, the numbers m and i have different meanings. In the symbol $\downarrow_{/m}$, the index m is a

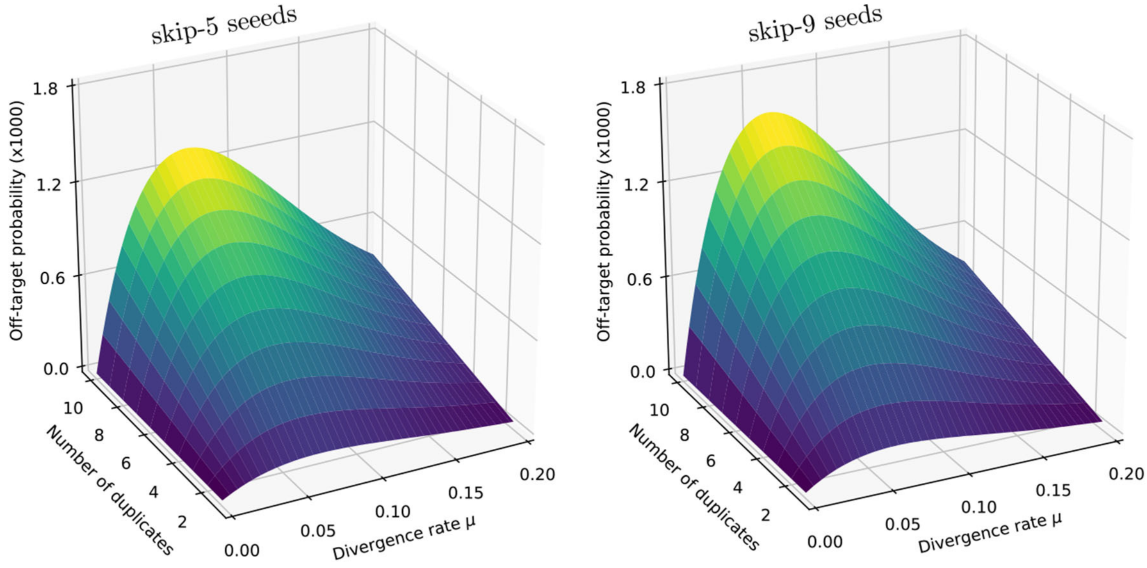


FIGURE 11 | Off-target seeding probabilities (skip seeds). The surfaces show the probability that seeding is off-target for skip-5 and skip-9 seeds of size $\gamma = 19$. Read size and error rate are the same as in **Figure 9**, i.e., $k = 50$ and $p = 0.01$. The divergence rate μ is defined as the probability that a given duplicate differs from the target at any given position. Note the factor 1,000 in the scale of the z-axis.

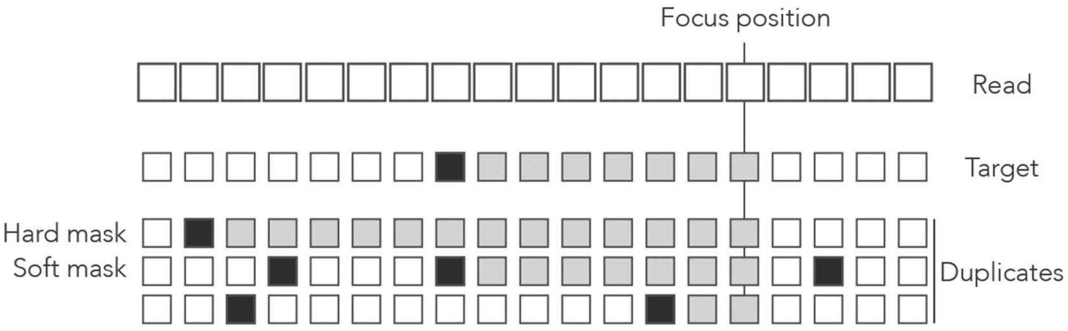


FIGURE 12 | Example of hard and soft masks. Genomic sequences are shown below a read where open squares represent nucleotides. In the sequences, the open squares represent matches and the closed squares represent mismatches. The nucleotides contributing to the match length are represented as gray boxes. At the focus position, the match length of the target is 7. The first duplicate is a hard mask because its match length is 13 > 7. The second duplicate is a soft mask because its match length is 7, as the target. The third duplicate is not a mask because its match length is 2 < 7.

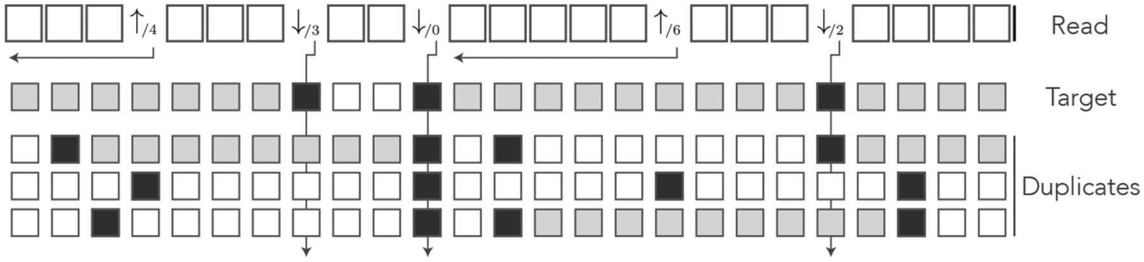


FIGURE 13 | The MEM encoding. The read of **Figure 3** is represented in the MEM alphabet. The arrows departing from the numbers help understand their meaning. The symbol $\downarrow_{/m}$ is indexed by the number m of sequences that match the nucleotide. The symbol $\uparrow_{/i}$ is indexed by the number i of nucleotides from the last error or from the beginning of the read. The gray squares in the symbolic sequences represent MEM seed matches. Other features are as in **Figure 8**.

number of sequences ($0 \leq m \leq N$ where N is the number of duplicates); in the symbol $\uparrow_{/i}$, the index i is a number of nucleotides. **Figure 13** shows the encoding of a read in the MEM alphabet.

The MEM alphabet captures the masking status of the nucleotide: the symbol $\downarrow_{/m}$ indicates that the nucleotide has m hard masks and $N - m$ soft masks. The symbols $\uparrow_{/i}$ indicate that the nucleotide is unmasked and that the previous nucleotide is masked.

In the MEM alphabet, strict on-target MEM seeds are the longest stretches of symbols containing some symbol $\uparrow_{/i}$ and not containing any symbol $\downarrow_{/m}$. Indeed, such a stretch is a match for the target because it does not contain any symbol $\downarrow_{/m}$, it only matches the target because it contains at least one unmasked nucleotide (marked by $\uparrow_{/i}$), and it cannot be extended because it is flanked by sequencing errors (symbols $\downarrow_{/m}$) or by the ends of the reads. Note that there is exactly one symbol $\uparrow_{/i}$ per strict on-target MEM seed, and therefore two symbols $\uparrow_{/i}$ must be separated by at least one symbol $\downarrow_{/m}$.

Shared on-target MEM seeds are the longest stretches of symbols $\square$ flanked by $\downarrow_{/0}$, or by the ends of the read. Indeed, such a stretch is a MEM seed because it matches the target and it cannot be extended ($\downarrow_{/0}$ is a mismatch against every sequence). Also, it cannot be a strict on-target MEM seed because it does not contain any $\uparrow_{/i}$ symbol, so it must be a shared on-target MEM seed.

As before, the read is converted from a sequence of symbols to a sequence of segments that consist of 0 or more symbols $\square$ followed by a terminator. We then specify the weighted generating functions of those segments and fill the transfer matrix $\mathring{M}_N(z)$ of the reads that do not contain an on-target MEM seed. We introduce the terms of the matrix by increasing order of complexity.

6.3. Segments Following $\uparrow_{/i}$

A segment terminated by $\uparrow_{/i}$ is the beginning of a strict on-target MEM of size at least i. The MEM reaches the next sequencing error or the end of the read, so the number of symbols $\square$ in the next segment must be at most $\gamma - i - 1$ and it must be terminated by a $\downarrow_{/m}$ symbol or by the tail terminator |.

The following definition will simplify the notations.

Definition 9. *Given the divergence rate μ and the number of duplicates N, the probability that a symbol is $\downarrow_{/m}$ given that the nucleotide is a read error is*

$$\omega_m = \binom{N}{m} \big(1 - \mu/3\big)^{N-m} \big(\mu/3\big)^m. \tag{16}$$

The expression for ω_m is exact: if the nucleotide is an error, the symbol is $\downarrow_{/m}$ for some m between 0 and N. Each duplicate is a match with probability $\mu/3$, so m has a Binomial distribution with parameters $(N, \mu/3)$.

On this segment, the matches between the read and the duplicates are irrelevant, so the weighted generating function of a symbol $\square$ is simply qz (recall that $p = 1 - q$ is the probability of a sequencing error). The weighted generating function of the terminator $\downarrow_{/m}$ is $\omega_m pz$, so the weighted generating function of the $\downarrow_{/m}$-segments following $\uparrow_{/i}$ is

$$D_{i,m}(z) = \omega_m pz \sum_{j=0}^{\gamma-i-1} (qz)^j. \tag{17}$$

And the weighted generating function of the tail segments following $\uparrow_{/i}$ is.

$$E_i(z) = \sum_{j=0}^{\gamma-i-1} (qz)^j. \tag{18}$$

6.4. Segments Following $\downarrow_{/m}$

The symbol $\downarrow_{/m}$ signifies that the nucleotide has m hard masks and $N - m$ soft masks. If all the masks vanish before the first read error, the next terminator will be a symbol $\uparrow_{/j}$, otherwise it will be the symbol | or a symbol $\downarrow_{/m}$. We separate the cases based on the terminator of the segment.

Case 1: The Terminator Is $\uparrow_{/j}$

Definition 10. *Given the divergence rate μ, the probability that a given duplicate contains a mismatch in a sequence of j error-free nucleotides is*

$$\xi_j = 1 - (1 - \mu)^j. \tag{19}$$

This is the probability that a hard or soft mask vanishes within j correct nucleotides.

The expression for ξ_j is exact: every nucleotide of the duplicate differs from the target with probability μ. In the absence of sequencing errors, this is also the probability that a nucleotide of the duplicate differs from the read. Given that there is no error, the probability that j nucleotides in a row are identical to the read is thus $(1 - \mu)^j$ and the probability that at least one of them is different is the complement $1 - (1 - \mu)^j$.

With this notation, the probability that at least one of N masks survives a sequence of j error-free nucleotides is thus $1 - (\xi_j)^N$, and the probability that there remains a mask at the $j - 1$-th but not at the j-th error-free nucleotide is $(\xi_j)^N - (\xi_{j-1})^N$. From this we conclude that the weighted generating function of the segments terminated by $\uparrow_{/j}$ following a segment terminated by $\downarrow_{/m}$ is

$$B_j(z) = \Big((\xi_j)^N - (\xi_{j-1})^N\Big)(qz)^j. \tag{20}$$

The fact that the reads have no on-target seeds imposes $j < \gamma$. Also note that this expression is the same for all symbols $\downarrow_{/m}$ (it does not depend on m).

Case 2a: The Terminator | Comes Before the γ-th Nucleotide

In this case there can be no on-target seed because the read finishes too early. However, we must enforce the condition that at least one of the N masks survives until the end, otherwise the segment would be terminated by one of the symbols $\uparrow_{/j}$. The weighted generating function is:

$$\sum_{i=0}^{\gamma-1} \Big(1 - (\xi_i)^N\Big)(qz)^i.$$

Case 2b: The Terminator | Comes After the γ-th Nucleotide

In this case, the soft masks do not hide the target. Even if a duplicate survives until the end of the read, there will be an on-target seed (shared in this case). To exclude on-target seeds, we must enforce the condition that at least one hard mask survives until the end of the segment (which is impossible if $m = 0$). The weighted generating function is

$$\sum_{i=\gamma}^{\infty}\Big(1-(\xi_i)^m\Big)(qz)^i.$$

Summing the expressions from cases 2a and 2b, we find that the weighted generating function of the tail following $\downarrow_{/m}$ is:

$$C_m(z)=\sum_{i=0}^{\gamma-1}\Big(1-(\xi_i)^N\Big)(qz)^i+\sum_{i=\gamma}^{\infty}\Big(1-(\xi_i)^m\Big)(qz)^i. \qquad (21)$$

Case 3a: The Terminator $\downarrow_{/n}$ Comes Before the γ-th Nucleotide

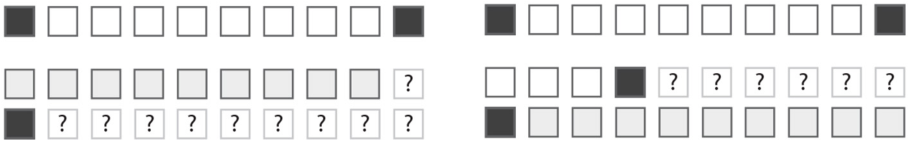
In this case, there can be no on-target seed and we must only exclude the terminators $\uparrow_{/j}$. As we have seen above, this implies that at least one of the N masks survives until the terminator. For a read of size $j+1$, this occurs with probability $1-(\xi_j)^N$. Including the terminator and summing over $j+1 \leq \gamma$, we see that the weighted generating function is:

$$\omega_n pz\sum_{j=0}^{\gamma-1}\Big(1-(\xi_j)^N\Big)(qz)^j. \qquad (22)$$

Case 3b: The Terminator $\downarrow_{/n}$ Comes After the γ-th Nucleotide

This case is by far the most convoluted. Since the segment contains at least γ error-free nucleotides, we must enforce the condition that it does not contain an on-target seed. This will be the case if any of the two following conditions is validated: (i) at least one hard mask covers all the error-free nucleotides, or (ii) all the hard masks vanish but at least one soft mask covers the whole segment (including the terminator).

The two conditions are mutually exclusive by construction. They are graphically represented in the diagram below. The left panel corresponds to case (i) and the right panel to case (ii). The top row represents the target, and the bottom rows represent duplicates (using the same symbols as in **Figure 13**).



Whenever a hard mask (here the first duplicate) covers the nucleotides as shown by the gray squares in the left panel, there can be no on-target seed. The positions marked with a question mark are irrelevant, they cannot change the fact that there is no on-target MEM seed. If the hard masks vanish, as in the right panel, then we need to look at the soft masks. If a soft mask covers the whole segment as indicated by the gray squares, then there can be no on-target seed. In all other cases there is an on-target MEM seed.

For a segment of size $j+1$, condition (i) has probability $\big(1-(\xi_j)^m\big)$. Summing over $j+1>\gamma$ and including the terminator, we see that the associated weighted generating function is

$$\omega_n pz\sum_{j=\gamma}^{\infty}\Big(1-(\xi_j)^m\Big)(qz)^j.$$

Condition (ii) is more convoluted, so we introduce some further notations to solve this sub-case.

Definition 11. *Given the divergence rate μ, the probability that a duplicate sequence contains a mismatch in a sequence of j error-free nucleotides followed by an error is*

$$\eta_j = 1-(1-\mu)^j\mu/3. \qquad (23)$$

This is the probability that a hard or soft mask vanishes within j correct nucleotides followed by a sequencing error.

The expression for η_j is exact: the probability that there are j matches between the duplicate and the target is $(1-\mu)^j$. If there are no sequencing errors, this is also the probability that there are j matches between the duplicate and the read. The probability that the duplicate matches the subsequent error is $\mu/3$, so the probability that there are $j+1$ matches including the sequencing error is $(1-\mu)^j\mu/3$. Finally, the probability that there is a mismatch is the complement $1-(1-\mu)^j\mu/3$.

Let us for now consider a segment of fixed size $j+1$. From expressions (19) and (23), the probability of condition (ii) is

$$(\xi_j)^m\Big(1-(\eta_j)^{N-m}\Big),$$

but we need to break up this term among all the possible terminators $\downarrow_{/n}$ $(0 \leq n \leq N)$ in order to fill the different entries of the transfer matrix. For this, we split this term in the number of soft masks that run until and including the terminator. From expression (23), the probability that there are $r \geq 1$ such soft masks is

$$\binom{N-m}{r}(1-\eta_j)^r(\eta_j)^{N-m-r}. \qquad (24)$$

For now we consider r fixed; we will compute the marginal probability at the final stage. By construction, each of those r soft masks matches the terminator, so the total number of matches is r plus the number of sequences that also match the terminator, among the remaining $N-m-r$ soft masks and the m hard masks.

Let us start with the m hard masks. The probability that each of them matches the terminator is simply $\mu/3$.

The case of the $N-m-r$ soft masks is more complicated because they can vanish precisely on the terminator—recall that in case (*ii*) all the hard masks are assumed to vanish before. If the soft mask failed within the first j nucleotides, then the $j+1$-th nucleotide can be anything and it will match the terminator with probability $\mu/3$. But if the soft mask survived the first j nucleotides, then it *must* fail on the $j+1$-th and it cannot match the terminator. From expressions (19) and (23), the probability that a given soft mask fails within the first j nucleotides is ξ_j/η_j—this is the conditional probability that it fails within the first j nucleotides given that it fails within the segment. Finally the probability that such a soft mask matches the terminator is $\mu/3 \cdot \xi_j/\eta_j$.

Summing the contributions of the hard masks (m in total, each matching the terminator with probability $\mu/3$) and of the soft masks ($N-m-r$ in total, each matching the terminator with probability $\mu/3 \cdot \xi_j/\eta_j$), the probability that the total number of matches is $n-r$ appears as the convolution product

$$\frac{(\mu/3)^{n-r}(1-\mu/3)^{N-n}}{(\eta_j)^{N-m-r}} \quad \psi_{j,m,n,r} \ ,$$

$$\text{where} \quad \psi_{j,m,n,r} = \sum_{q\geq 0}\binom{m}{q}\binom{N-m-r}{n-r-q}(\xi_j)^{n-r-q}.$$

Finally, we need to compute the marginal probability over the number r of soft masks that survive until the end of the read. Multiplying by the probability of r from expression (24) and summing over $r \geq 1$, the probability that n duplicate sequences match the terminator appears as

$$\begin{aligned}
&\sum_{r\geq 1}\binom{N-m}{r}(1-\eta_j)^r(\mu/3)^{n-r}(1-\mu/3)^{N-n}\psi_{j,m,n,r} \\
&= (\mu/3)^n(1-\mu/3)^{N-n}\sum_{r\geq 1}\binom{N-m}{r}(1-\mu)^{rj}\psi_{j,m,n,r} \\
&= \omega_n \cdot \zeta_{j,m,n},
\end{aligned}$$

where

$$\zeta_{j,m,n} = \sum_{r\geq 1}\binom{N-m}{r}(1-\mu)^{rj}\psi_{j,m,n,r}\bigg/\binom{N}{n}. \tag{25}$$

This is the probability that the terminator is the symbol $\downarrow_{/n}$ given that the segment has size $j+1 > \gamma$, that the first sequencing error occurs on the last nucleotide, that the preceding terminator was $\downarrow_{/m}$ and that the m hard masks fail before the end of the segment.

Summing the terms from case 3a and from case 3b, we find that the weighted generating function of the $\downarrow_{/n}$ segments following $\downarrow_{/m}$ is:

$$\begin{aligned}
A_{m,n}(z) = \omega_n pz \sum_{j=0}^{\gamma-1}\Big(1-(\xi_j)^N\Big)(qz)^j \\
+\omega_n pz \sum_{j=\gamma}^{\infty}\Big(1-(\xi_j)^m \cdot (1-\zeta_{j,m,n})\Big)(qz)^j.
\end{aligned} \tag{26}$$

6.5. Transfer Matrix

Collecting and arranging the results above, we can verify that the final expression of the transfer matrix $\mathring{M}_N(z)$ is

$$\begin{array}{c|ccccccc}
 & \downarrow_{/0} & \cdots & \downarrow_{/N} & \uparrow_{/1} & \cdots & \uparrow_{/\gamma-1} & | \\
\hline
\downarrow_{/0} & A_{0,0}(z) & \cdots & A_{0,N}(z) & B_1(z) & \cdots & B_{\gamma-1}(z) & C_0(z) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\downarrow_{/N} & A_{N,0}(z) & \cdots & A_{N,N}(z) & B_1(z) & \cdots & B_{\gamma-1}(z) & C_N(z) \\
\uparrow_{/1} & D_{1,0}(z) & \cdots & D_{1,N}(z) & 0 & \cdots & 0 & E_1(z) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\uparrow_{/\gamma-1} & D_{\gamma-1,0}(z) & \cdots & D_{\gamma-1,N}(z) & 0 & \cdots & 0 & E_{\gamma-1}(z) \\
| & 0 & \cdots & 0 & 0 & \cdots & 0 & 0
\end{array}$$

where

$$\begin{aligned}
A_{m,n}(z) =& \omega_n pz \sum_{i=0}^{\gamma-1}\Big(1-(\xi_i)^N\Big)(qz)^i \\
&+\omega_n pz \sum_{i=\gamma}^{\infty}\Big(1-(\xi_i)^m \cdot (1-\zeta_{i,m,n})\Big)(qz)^i
\end{aligned} \tag{26}$$

$$B_i(z) = \Big((\xi_i)^N - (\xi_{i-1})^N\Big)(qz)^i \tag{20}$$

$$C_m(z) = \sum_{i=0}^{\gamma-1}\Big(1-(\xi_i)^N\Big)(qz)^i + \sum_{i=\gamma}^{\infty}\Big(1-(\xi_i)^m\Big)(qz)^i \tag{21}$$

$$D_{j,m}(z) = \omega_m pz \sum_{i=0}^{\gamma-j-1}(qz)^i \tag{17}$$

$$E_j(z) = \sum_{i=0}^{\gamma-j-1}(qz)^i \tag{18}$$

and where

$$\omega_m = \binom{N}{m}\Big(1-\mu/3\Big)^{N-m}\Big(\mu/3\Big)^m \tag{16}$$

$$\xi_j = 1-(1-\mu)^j \tag{19}$$

$$\eta_j = 1-(1-\mu)^j\mu/3 \tag{23}$$

$$\zeta_{j,m,n} = \sum_{r\geq 1}\binom{N-m}{r}(1-\mu)^{rj}\psi_{j,m,n,r}\bigg/\binom{N}{n} \tag{25}$$

$$\psi_{j,m,n,r} = \sum_{q\geq 0}\binom{m}{q}\binom{N-m-r}{n-r-q}(\xi_j)^{n-r-q}.$$

Remark 4. *In the special case $N = 0$, the transfer matrix simplifies to the extent that we can compute the weighted generating function of the reads without on-target MEM seed in closed form. The result is*

$$\frac{1 + qz + \ldots + (qz)^{\gamma-1}}{1 - pz\big(1 + qz + \ldots + (qz)^{\gamma-1}\big)}. \qquad (2)$$

Expression (2) was shown in section 3.4 to be the weighted generating function of reads without on-target exact seed. This shows that when there are no duplicates, MEM seeds have exactly the same properties as exact seeds.

6.6. Computing MEM Seeding Probabilities

The matrix $\mathring{M}_N(z) \cdot (I - \mathring{M}_N(z))^{-1}$ contains the weighted generating functions of all the reads without on-target MEM seeds. The term of interest, as usual, is the top right entry. Indeed, every read can be prepended by $\downarrow_{/0}$-segments and only by those, otherwise the read would start with fewer than N soft masks. Thus, reads without an on-target MEM seed are precisely the sequences of segments that can be appended to the symbol $\downarrow_{/0}$ and that are terminated by a tail.

To compute this term, we proceed as in section 3.6, i.e., we compute the powers of $\mathring{M}_N(z)$ in the arithmetic of truncated polynomials and we stop the iterations when the terms are negligible. We bound the probability that the read contains more than e sequencing errors using the expression

$$Pr(X \geq e) \leq \exp\left((e-k)\log\frac{k-e}{k(1-p)} - e\log\frac{e}{kp}\right), \qquad (7)$$

but here not every segment contains an error. There cannot be two symbols $\uparrow_{/j}$ in a row, so a read with $s+1$ segments must contain a minimum number of sequencing errors which is $e = \lfloor s/2 \rfloor$. As before, we compute the powers of $\mathring{M}_N(z)$ until the upper bound is less than a set fraction ε of the current value of a_k.

If we call M_0 the event that the read contains an on-target MEM seed, the method above gives us $P(\overline{M}_0)$. Calling M_j the event that the read contains a MEM seed for the j-th duplicate, we are interested in the probability

$$P\big(\overline{M}_0 \cap (M_1 \cup \ldots \cup M_N)\big) = P(\overline{M}_0) - P\big(\overline{M}_0 \cap \overline{M}_1 \cap \ldots \cap \overline{M}_N\big). \quad (8)$$

The key insight to compute $P\big(\overline{M}_0 \cap \overline{M}_1 \cap \ldots \cap \overline{M}_N\big)$ is to realize that there is some MEM seed, on-target or not, if and only if the read contains a match of size γ or more for any of the $N+1$ sequences. Therefore, this probability is the same as the term $P\big(\overline{S}_0 \cap \overline{S}_1 \cap \ldots \cap \overline{S}_N\big)$ computed in section 4.

In conclusion, the probability that the MEM seeding process is off-target is

$$P(\overline{M}_0) - P(\overline{S}_0) \cdot \left(\frac{P(\overline{S}_0 \cap \overline{S}_1)}{P(\overline{S}_0)}\right)^N, \qquad (27)$$

where $P(\overline{M}_0)$ is computed using $\mathring{M}_N(z)$ as explained in this section, $P(\overline{S}_0)$ is computed using a recursive equation as explained in section 3.4, and $P(\overline{S}_0 \cap \overline{S}_1)$ is computed using $\tilde{M}_0(z)$ as explained in section 4.1.

6.7. Monte Carlo Sampling

One potential difficulty in computing $P(\overline{M}_0)$ is that the matrix $\mathring{M}_N(z)$ has dimension $(N+\gamma+1) \times (N+\gamma+1)$. The problem can become computationally intractable because N can be very large. For instance, the sequences called *Alu* have more than one million duplicates in the human genome. There is no hope to compute the powers of $\mathring{M}_N(z)$ in these conditions and we need an alternative method.

The symbolic representation as MEM segments can be used to design an efficient method to sample reads. Instead of generating the nucleotides of the $N+1$ sequences one by one, we can generate a single sequence of segments. Since the number of segments does not depend on N, we can obtain a fast Monte Carlo method to sample millions of reads and count the proportion that contain an on-target MEM seed.

The principle is to proceed in cycles of two steps. We first sample the position of the next sequencing error, which gives the position of the next symbol $\downarrow_{/m}$, where m will be determined at a later stage. The second step is to determine whether there is a symbol $\uparrow_{/j}$ before that. For this we sample the number of masks that vanish before the symbol $\downarrow_{/m}$. If they all vanish, the read contains an on-target MEM seed, provided the next read error is at a distance greater than γ. Otherwise, we sample the number m of hard masks at the sequencing error, and the process is repeated until we generate an on-target MEM seed, or until the read has size k or greater (in which case it has no on-target MEM seed).

The method is summarized in algorithm 1 below. It requires efficient algorithms to sample from the geometric and from the binomial distributions. Sampling from a geometric distribution can be done by computing the logarithm of a uniform $(0,1)$ random variable. Sampling from a binomial distribution can be done by the method of Kachitvichyanukul and Schmeiser (1988). Most importantly, the number of duplicates N has little influence on the running speed of algorithm 1.

Algorithm 1 is an important result. It gives a compact solution to the problem of estimating the probability that a read can be mapped when using MEM seeds. The algorithm is also much faster than the naive approach of sampling every nucleotide of every sequence, because it is equivalent to sampling the nucleotide sequence of *all* the duplicates.

6.8. Illustration

We illustrate the strategy delineated above using the same settings as in section 4.6 ($k = 50$, $p = 0.01$, and $\gamma = 19$), except that we replace exact seeds by MEM seeds.

Figure 14 shows the result for a number of duplicates N from 1 to 10, for a divergence rate μ from 0 to 0.20 and for MEM seeds of different minimum size γ. The surfaces have the same general aspect as those of **Figure 9**. The probability that seeding is off-target increases with N, as shown by expression (27).

There is again a worst value of μ in each plot, but it is much lower than the previous two cases (it is close to 0.028 for $\gamma = 14$ and $\gamma = 19$, 0.030 for $gamma = 27$ and 0.035 for $\gamma = 34$). In the case of MEM seeds, it is not obvious why the same value of μ maximizes (27) for all values of N because both $P(\overline{M}_0)$ and $P(\overline{S}_0 \cap \overline{S}_1)$ depend on μ.

```
Parameter: $k$ is the size of the reads.
Parameter: $p$ is the error rate of the sequencer
           (substitutions only).
Parameter: $N$ is the number of duplicates.
Parameter: $\mu$ is the nucleotide-wise probability that
           duplicates differ.
Result: Sample a read at random. Return 1 if the read
        contains a good MEM seed, otherwise return 0.
$\lambda \leftarrow 0$ ; ▷ Current read size.
$m \leftarrow 0$ ;     ▷ Current number of hard masks.
while $\lambda < k$ do
    $i \leftarrow geom(p) - 1$ ; ▷ Error-free nucleotides.
    if $i \geq k - \lambda$ then
        if $k - \lambda < \gamma$ then
            return 0;
        else
            $h \leftarrow binom(m,(1-\mu)^{k-\lambda})$   ▷Surviving hard masks.
            return 1 if $h = 0$, otherwise return 0;
        end
    else
        $h \leftarrow binom(m,(1-\mu)^{i})$            ▷Surviving hard masks.
        $s \leftarrow binom(N-m,(1-\mu)^{i}\mu/3)$     ▷Surviving soft masks.

        if $i \geq \gamma$ and $h = 0$ and $s = 0$ then
            return 1;
        else
            $m \leftarrow s + binom(N - s, \mu/3)$;
            $\lambda \leftarrow \lambda + i + 1$;
        end
    end
end
```

Figure 14 reveals that in this concrete case, MEM seeds increase the chances that seeding is off-target by a factor 15 compared to exact seeds (see **Figure 9**). On this criterion, MEM seeds are always inferior to exact seeds with the same specifications. The reason is that a MEM seed always contains at least one exact seed of size γ. MEM seeds tremendously simplify the seeding process, but this comes at the cost of an increase of the probability that seeding is off-target.

Also note that the values change less than one might expect when varying the minimum seed length γ. Part of the reason is that MEM seeds are typically longer than their minimum size γ. In any event, this example shows that reducing γ to gain sensitivity provides very modest benefits, with potentially high costs in terms of short spurious hits.

7. THE SESAME LIBRARY

We implemented the methods and algorithms presented here in an open-source C library to compute seeding probabilities. The library is called Sesame and is available at https://github.com/gui11aume/sesame.

7.1. Main Features of Sesame

Sesame contains functions to compute the seeding probabilities described here. All the functions were tested against simulations to ensure that the implementation is accurate and the code was checked extensively by static analysis and unit testing.

Computing seeding probabilities can take up to a few seconds, even when replacing iterative methods by Monte Carlo simulations. This is incompatible with the speed requirements of modern mappers, so Sesame has an interface for this type of application. In this mode, Sesame computes the results only the first time and stores them in memory for reuse on subsequent calls.

Storing the results in memory is an efficient strategy because three parameters are constant throughout the sequencing run: the minimum seed size γ, the read size k and the error rate of the sequencer p (and for skip seeds, the number of skipped nucleotides n is also constant). Only two parameters depend on the read: the number of duplicates N and their divergence rate μ. In this mode, Sesame automatically switches to Monte Carlo sampling when N is large to save time. Also, the input parameters are "snapped" to a predefined grid of set values for N and μ, so that few computations are performed and most of the calls are actually memory lookups. Sesame can thus be integrated in short read mappers without being a bottleneck.

Alternatively, the probabilities of interest can be computed offline, saved to disk and loaded at run time. This is particularly useful if the sequencing runs follow some standard conditions with a known error rate, because the computations can be recycled between runs.

Finally, Sesame also has an offline interface, where seeding probabilities are computed exactly as requested by the users, i.e., without modifying the algorithm or the parameters, and also without storing the results in memory.

The Sesame manual, available from the repository, contains additional information and explains in detail how to use the library.

7.2. Using Sesame to Compare Seeding Strategies

As a tool to compute seeding probabilities, Sesame can be used to compare the merits of different strategies. The kind of insight that we can gain from such calculations was already showcased in **Figures 9**, **11**, **14**, where the numbers were computed using Sesame.

To further showcase the potential benefits of computing seeding probabilities, we use Sesame to compare the default seeding strategies of BWA-MEM (Li, 2013) and Bowtie2 (Langmead and Salzberg, 2012). Note that both mappers use advanced techniques to refine the seeds, so this comparison does not reflect the true performance of the mappers. It is nevertheless useful to know the baseline of each strategy. The default of BWA-MEM is to use MEM seeds of minimum size 19; that of Bowtie2 is to use skip-9 seeds of size 16.

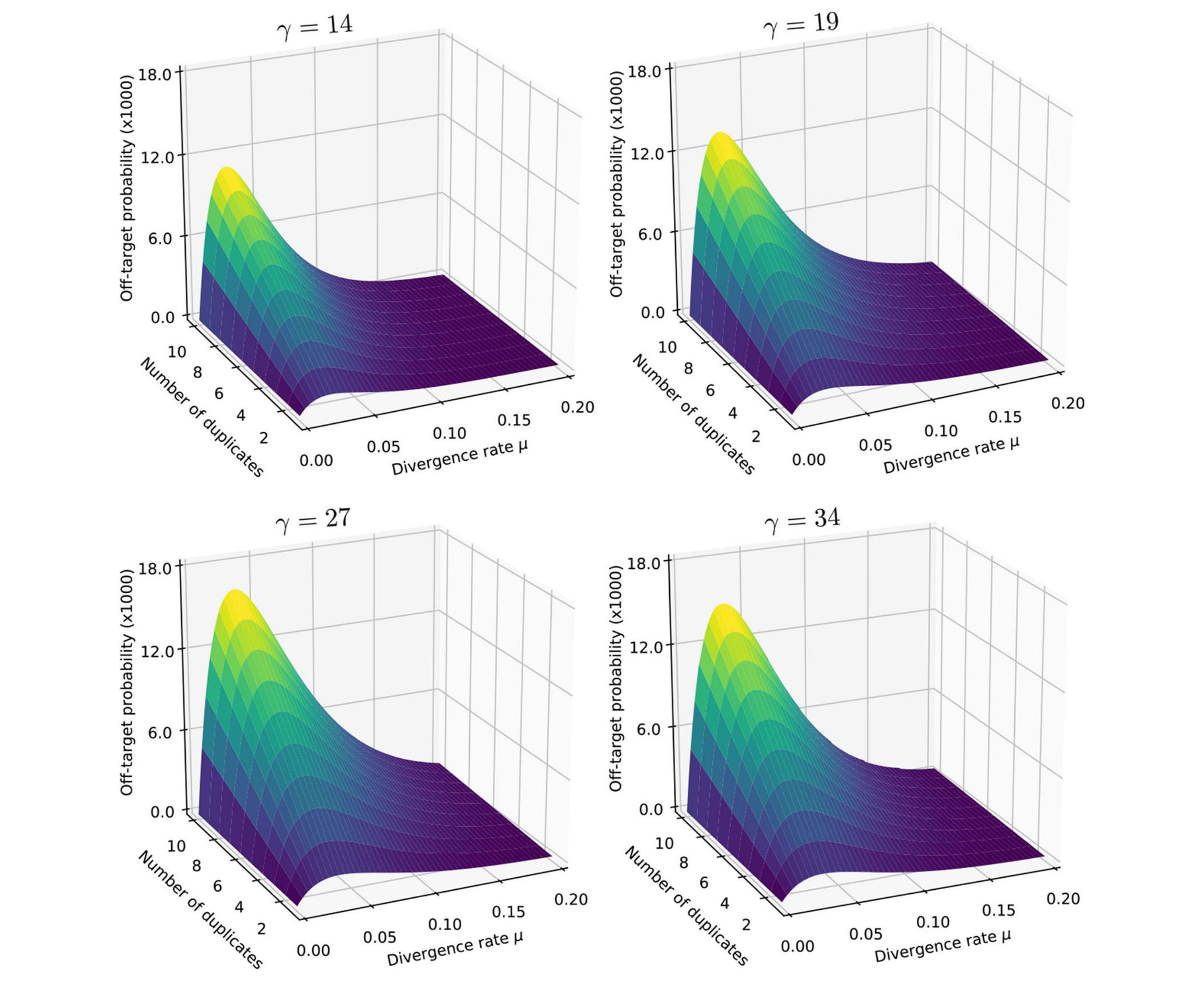


FIGURE 14 | Off-target seeding probabilities (MEM seeds). The surfaces show the probability that seeding is off-target for MEM seeds of indicated minimum size γ. Read size and error rate are the same as in **Figure 9**, i.e., $k = 50$ and $p = 0.01$. The divergence rate μ is the probability that a given duplicate differs from the target at any given position. Note the factor 1,000 in the scale of the z-axis.

The probabilities that seeding is off-target for different read sizes k and different number of duplicates N are plotted in **Figure 15**. The left panel shows the results for MEM seeds and the right panel shows the results for skip seeds. Here the error rate p is set to 1%, close to the specifications of the Illumina platform (Nakamura et al., 2011), and the divergence rate between duplicates μ is set to an arbitrary value of 6%.

The behaviors of the two types of seeds are dramatically different. Let us start with MEM seeds. For every value of N, the probability initially increases with the read size, and then drops exponentially. The initial increase is a hallmark of MEM seeds; it is due to the fact that duplicates can mask the target. Note that the asymptotic decay depends on the number of duplicates N because each duplicate can mask the target and thereby reduces the probability that it is discovered. Overall, these results show that the performance of MEM seeds is poor when the target has more than approximately 20 duplicates.

Turning to skip seeds, we see that the curves have a staircase look with a drop every 10 nucleotides. This is so because seeding probabilities remain unchanged until there is space for another seed of size 16 on the read. The curves are otherwise decreasing with a steady exponential trend where the asymptotic decay does not depend on the number of duplicates N. The reason is that duplicates do not prevent the target from being discovered, they merely fool the mapper when the target was not found. This property makes the asymptotic decay of skip seeds substantially faster that of MEM seeds when N is large.

Those properties are self-evident in retrospect, but they are not necessarily obvious from the definitions of MEM seeds and skip seeds. The main limitations of the MEM seeds are best

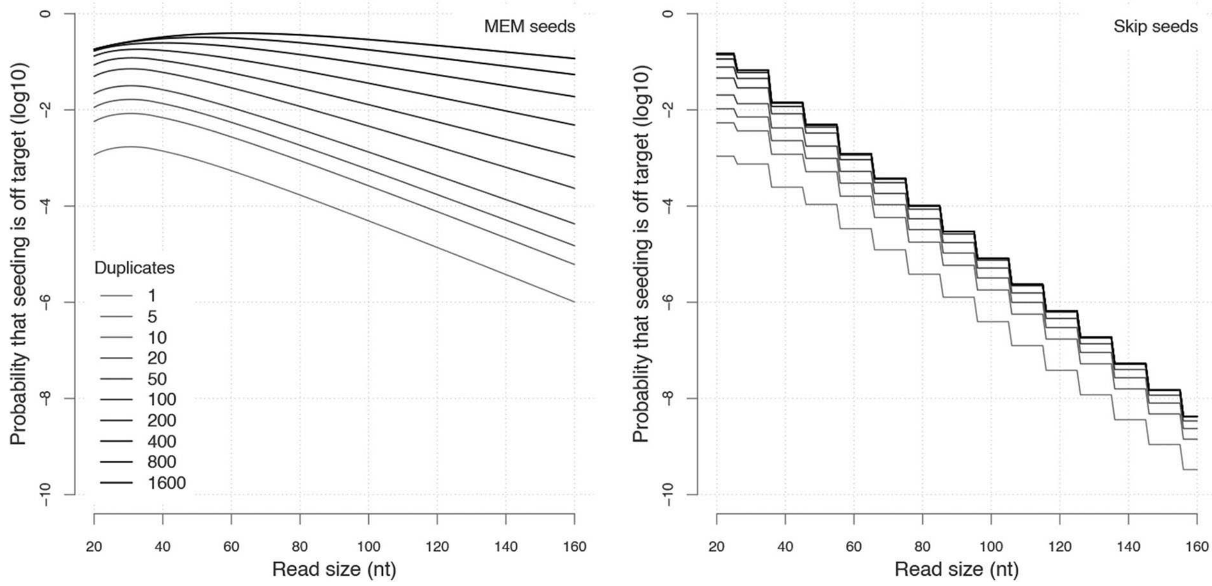


FIGURE 15 | Comparing seeding strategies with Sesame. **(Left)** MEM seeds with $\gamma = 19$ and $\mu = 0.06$. **(Right)** Skip-9 seeds with $\gamma = 16$ and $\mu = 0.06$. The probabilities that seeding is off-target were computed using Sesame. Each curve represents the probability for a given number of duplicates (*N*). Estimates using iterative methods (MEM seeds where $N \leq 20$) were computed to within 1% accuracy. Estimates using Monte Carlo sampling (MEM seeds where $N > 20$) were computed as the average of 500 million simulations.

understood by keeping in mind that their asymptotic decay depends on *N*.

Figure 15 suggests that skip-9 seeds of size 16 are just better than MEM seeds of size 19. However, the gain in sensitivity comes at the cost of a larger candidate set, slowing down the mapping process. To remain competitive, Bowtie2 further filters the candidate set using a priority policy. But since some candidates are not checked, the probability that seeding is off-target is larger than shown in **Figure 15**. Also, we will show in section 8.1 that the higher sensitivity of skip seeds is not as big an advantage as it looks because seeding is not the only source of mapping errors.

7.3. Key Insights About MEM Seeds

At least two key insights about MEM seeds can be gained from **Figure 15**. The first is that it is not worth it for a MEM-based mapper to check all the candidate loci when there are more than approximately 20 of them. The mapper may find the correct location, but even if this is the case, the mapping quality will remain low because the prior chances of failure were high. A better strategy is to either bail out to not waste time, or to switch to a more sensitive seeding method (BWA opts for the second and uses a re-seeding policy). It is also important to note that this decision should be based on an estimated value of *N* and not, for instance, on the size of the seed or some other variable.

A second insight is that for MEM seeds, the off-target rate is always above 10^{-3} for reads of 50 nucleotides or fewer. Here it is important to mention that the value $\mu = 0.06$ is not even the worst for reads of this size when $p = 0.01$ (according to **Figure 14**, the worst value is around 0.02–0.03). So if μ is unknown and one wishes to be conservative, it seems that reads of 50 nucleotides cannot be mapped with confidence better than $1/1,000$.

However, this is not true for the reason that it is practically impossible to map an Illumina read of size 50 to the wrong location when $N = 0$ (see section 8.3). Indeed, incorrect locations are unrelated sequences in this case, and it is easy to recognize that two sequences of size 50 are not homologous. This means that the highest impact one can have on the mapping quality is to check whether *N* is 0, or in other words, whether the target is a unique sequence.

These insights suggest that there is a way to make MEM seeds more useful for short read mappers. Following these principles, we have implemented a prototype mapper based on Sesame that shows good overall performance (Zorita et al., 2020).

8. PRACTICAL CONSIDERATIONS

8.1. True vs. Best Location

Early in the development of the theory, we distinguished the true location from the best location. We swiftly assumed that they are identical in order to eliminate some practical considerations that would otherwise clutter the exposition. Throughout the article we have developed a framework to compute the probability that the *true* location is in the candidate set. We have not mentioned anything about the probability that the *best* location is in the candidate set, so our results do not address the best location problem. It remains to establish how they contribute to addressing the *true* location problem.

The issue at hand is that the candidates are tested with an alignment algorithm that returns the best location. So even if the true location is in the candidate set, the read may be

TABLE 1 | Simulations of mapping errors with MEM seeds.

k	*N*	Not seeded	Seeded, not best	Relative error
50	1	$9.2 \cdot 10^{-4}$	$8.4 \cdot 10^{-5}$	1.09
50	10	$9.0 \cdot 10^{-3}$	$7.9 \cdot 10^{-4}$	1.09
50	100	$8.1 \cdot 10^{-2}$	$5.1 \cdot 10^{-3}$	1.06
100	1	$2.4 \cdot 10^{-5}$	$7.3 \cdot 10^{-6}$	1.30
100	10	$2.9 \cdot 10^{-4}$	$6.9 \cdot 10^{-5}$	1.24
100	100	$5.5 \cdot 10^{-3}$	$5.7 \cdot 10^{-4}$	1.10

Ideal reads were randomly generated 1,000,000 times with sequencing error rate $p = 0.01$, divergence rate $\mu = 0.06$, varying the read size k and number N of duplicates as indicated, using MEM seeds of minimum size $\gamma = 19$. Not seeded shows how often the read had no seed for the target, Seeded, not best shows how often the read had a seed for the target but the target was not the best candidate, and Relative error shows the factor between the probability that the read is not mapped to the true location and the probability that seeding is off-target.

TABLE 2 | Simulations of mapping errors with MEM seeds.

k	*N*	Not seeded	Seeded, not best	Relative error
50	1	$1.7 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$	3.9
50	10	$1.6 \cdot 10^{-3}$	$4.7 \cdot 10^{-3}$	3.9
50	100	$7.9 \cdot 10^{-3}$	$4.4 \cdot 10^{-2}$	6.6
100	1	$4.4 \cdot 10^{-8}$	$2.3 \cdot 10^{-5}$	523.7
100	10	$3.5 \cdot 10^{-7}$	$2.3 \cdot 10^{-4}$	658.1
100	100	$1.0 \cdot 10^{-6}$	$2.3 \cdot 10^{-3}$	2301.0

*Ideal reads were randomly generated 1,000,000 times with sequencing error rate $p = 0.01$, divergence rate $\mu = 0.06$, varying the read size k and number N of duplicates as indicated, using skip-9 seeds of size $\gamma = 19$. The meanings of the columns are the same as in **Table 1**.*

TABLE 3 | Simulations of seeding errors with MEM seeds.

k	*N*	Sesame	Simulation	Ratio
50	1	$4.5 \cdot 10^{-4}$	$3.8 \cdot 10^{-4}$	1.18
50	10	$4.5 \cdot 10^{-3}$	$4.4 \cdot 10^{-3}$	1.02
50	100	$4.0 \cdot 10^{-2}$	$4.2 \cdot 10^{-2}$	0.95
100	1	$3.7 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	1.05
100	10	$4.2 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$	1.02
100	100	$8.9 \cdot 10^{-3}$	$8.6 \cdot 10^{-3}$	1.03

Realistic Illumina HiSeq 2,000 reads were randomly generated 1,000,000 times varying the read size k and number N of duplicates as indicated, and fixing the divergence rate to $\mu = 0.06$ and using MEM seeds of minimum size $\gamma = 19$. Sesame shows the probability that seeding is off-target as computed with Sesame, Simulation shows this probability estimated by the simulation, and Ratio shows their ratio.

mapped somewhere else because some other candidate has a better alignment score. The question is how often this happens. If this is a rare event, our results give a good approximation of the probability that a read is mapped to the true location. Otherwise the estimates may not be so useful in practice.

The answer depends on the seed type. We start with MEM seeds because they will allow us to highlight an important phenomenon. The key insight is that MEM seeds tend to exclude the target from the candidate set if it is not the best location. To develop an intuition as to why this is the case, consider a read with a single sequencing error. The only way a candidate can have a better score is to be a perfect match for the read. But in this case the true location is hard-masked (see definition 8) and it is not in the candidate set.

Excluding ties, there are two possibilities: either the true location is the best, in which case it is seeded and the read is correctly mapped; or the true location is not the best, in which case it is not seeded and the read is mapped incorrectly. The read is mapped to the true location if and only if seeding is on-target. In practice there are ties, and reads can have more than one error, but this is a general trend for MEM seeds.

Table 1 shows the results of simulations with MEM seeds where we dissociate the true from the best location. In each tested condition, we record how often the reads are mapped incorrectly because seeding was off-target. Importantly, we also record the cases where reads are mapped incorrectly despite the fact that seeding was on-target. The results show that the second case occurs a minor fraction of the time, meaning that the probability that seeding is on-target is close to the probability that the read is mapped to the true location.

For exact and skip seeds, the conclusions are different. The situation is even the opposite if we consider reads with a single sequencing error. In this case the target is always in the candidate set (if the read size is greater than 2γ, where γ is the minimum seed length), so mapping errors are never due to failures of the seeding heuristic. Instead, mapping errors happen only when the true location is not the best.

Table 2 shows the results of similar simulations with skip-9 seeds. In all the tested cases the probability that the true location is not the best is substantially higher than the probability that seeding is off-target. As a result, the Sesame estimates are very far from the probability that the read is not mapped to the true location. The results for exact seeds are omitted, but they are qualitatively similar to those obtained for skip-9 seeds.

In conclusion, the off-target probabilities computed by Sesame are close to the probability that the read is not mapped to the true location when using MEM seeds. In contrast, the estimates are very far when using exact seeds and skip seeds. In this case, it is more accurate to use an estimate of the probability that the true location is not the best, which is relatively easy to compute given the error rate of the sequencer p, the number of duplicates N and their divergence rate μ.

8.2. Realistic Sequencing Errors

In order to make the model tractable, we had to make some simplifying assumptions regarding the distribution of errors (section 3.1). In particular, we assumed that sequencing errors are uniform on the read, which is known to not be the case with current instruments. The errors are typically more frequent at the ends of the reads (Nakamura et al., 2011), meaning that seeds may be longer than expected, impacting the probability that seeding is off-target.

To test whether this is the case, we used the ART simulator (Huang et al., 2012) to emulate the error distribution of Illumina reads as per technology standards of 2016. We used the settings for the Illumina HiSeq 2000 and generated random reads of size 50 and 100 from the human genome. We estimated the probability that seeding is off-target from simulations and compared the results to calculations performed by Sesame (ART

TABLE 4 | Simulations of seeding errors with skip-9 seeds.

k	N	Sesame	Simulation	Ratio
50	1	$4.5 \cdot 10^{-5}$	$4.5 \cdot 10^{-5}$	1.00
50	10	$4.2 \cdot 10^{-4}$	$3.9 \cdot 10^{-4}$	1.07
50	100	$2.1 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	1.11
100	1	$< 10^{-6}$	$< 10^{-6}$	NA
100	10	$3 \cdot 10^{-6}$	$2 \cdot 10^{-5}$	1.50
100	100	$8 \cdot 10^{-6}$	$6 \cdot 10^{-4}$	1.33

*Realistic Illumina HiSeq 2,000 reads were randomly generated 1,000,000 times varying the read size k and number N of duplicates as indicated, and fixing the divergence rate to $\mu = 0.06$ and using MEM seeds of minimum size $\gamma = 19$. The meanings of the columns are the same as in **Table 3**.*

suggests error rates $p = 0.0052$ and $p = 0.0075$ for reads of size 50 and 100, respectively).

Table 3 shows the results for MEM seeds of minimum size $\gamma = 19$. The discrepancy with Sesame estimates is always within a factor 1.2. **Table 4** shows the same type of comparison for skip-9 seeds of size $\gamma = 19$. The values are substantially lower than for MEM seeds because skip-9 seeds are more sensitive. In these conditions, the discrepancy with the Sesame estimates is within a factor 1.5, but it is important to highlight that simulations are unreliable for events with very low probability.

Overall, these results suggest that the relatively strong assumptions regarding the distribution of errors are not a major issue in practice. The answer of course depends on the type of sequencer that is used and on the particulars of the mapping problem.

In practical applications, it is likely that other factors will be more detrimental for the precision of the estimates. For instance, the number of duplicates N and their divergence rate μ are unknown, introducing an uncertainty that propagates to the Sesame estimates. Importantly, we assumed that duplicates evolve independently of each other and with uniform substitutions even in the simulations featured in **Tables 3**, **4** because more realistic models are tedious to implement. The burden of this assumption on the precision of the estimate in practical settings is difficult to gauge. Overall, Sesame estimates should be considered approximate in all applications with real biological data.

8.3. Spurious Random Hits

We have assumed that seeds can match only the target or one of its duplicates. In reality, seeds have a small chance to match any sequence of the genome.

We have assumed throughout that all the genomic sequences that have a perfect match for a seed are considered candidate locations. The hope is that spurious random matches are shorter than the minimum seed length γ, but this is not always the case. And as mentioned above, every candidate location must be verified by an exact alignment algorithm.

Such random hits can be problematic for two reasons: First, the time spent verifying them is wasted. Second, they can cause false positives. Fortunately, both issues can be addressed.

To save time during the alignment phase, we can prioritize the seeds so that the best location is likely to be discovered first, allowing us to bail out from other alignments as early as possible. In some cases, it is even possible to give an upper bound on the alignment score of a candidate location so that the alignment can be skipped altogether. These considerations depend on the implementation of the mapper so we will not develop this further. What matters is that random hits do not impose a significant burden on the time needed to verify the candidates.

The second issue is that random hits can generate false positives. Such cases occur when seeding is null (meaning that there is no seed for either the target or any of its duplicates). Even when seeding is null, the candidate set is in general not empty because of spurious random hits. But since such hits are not homologous to the read, the alignment score is noticeably low, and this case is easy to detect.

Indeed, if the candidate location is random, mismatches occur with probability 3/4. If it is the true location, mismatches occur with probability p. We discard the seed because it is an automatic match of size γ (and possibly larger in the case of MEM seeds). Say that there remain L nucleotides and that m of them are mismatches for the candidate location. From Bayes formula, the probability that the location is random given the number of mismatches is

$$\frac{1}{1 + 4^L (p/3)^m q^{L-m} (1-\beta)/\beta}, \tag{28}$$

where $q = 1 - p$ and β is the prior probability that the hit is spurious.

The value of β has little importance if p is small. To give an example, say that $p = 0.01$ and that a read generates a hit in the genome such that 33 nucleotides need to be aligned after seeding. If the hit is random there is a 99.99% chance that at least 15 are mismatches. For $m = 15$, the denominator of expression (28) is approximately $1 + 4.3 \cdot 10^{-18}(1-\beta)/\beta$. So unless $\beta < 10^{-17}$, the support for the hypothesis that the hit is random is overwhelming. Conversely, if the hit is not random, there is a 99.99% chance that 4 or fewer nucleotides are mismatches. For $m = 4$, the denominator is approximately $1 + 6.8 \cdot 10^9(1-\beta)/\beta$, so unless $1 - \beta < 10^{-9}$, the support for the hypothesis that the hit is not random is overwhelming. In summary, if p is small, we do not need to worry about the value of β, one can choose for instance $\beta = 1/2$ so that the term $(1-\beta)/\beta$ disappears from expression (28).

Since the probability that the best candidate is a random sequence is either very small or very large, it has no influence in the first case, and it dominates the probability of a false positive in the second case. Checking the value of expression (28) after mapping reveals whether the hit should be discarded and the read should be considered unmapped.

In summary, spurious random hits occur regularly, but it is possible to minimize their computational burden. Also, they are no cause for concern regarding false positives because they can easily be detected using Bayes' formula, as shown in expression (28).

9. DISCUSSION

We have devised a set of methods to compute the probability that seeding heuristics fail and commit the mapping process to an error. We have also implemented the algorithms as an open source C library to perform the computations. This fills a knowledge gap to understand and calibrate the performance of the seeding heuristics. The pillars of our strategy are borrowed from analytic combinatorics (Régnier, 2000; Nicodeme et al., 2002; Flajolet and Sedgewick, 2009; Sedgewick and Flajolet, 2013), even though we do not follow the complete programme. Constructing generating functions usually serves the purpose of finding their singularities in order to approximate the solution. In our case, however, the weighted generating functions cannot be computed so this strategy is not applicable.

To find the probabilities of interest in the absence of a fully specified weighted generating function, we compute only the first terms of the Taylor series using iterative methods as explained in section 3.6, or using Monte Carlo sampling as explained in section 6.7. In this regard, the breakthrough is the encoding of reads as segments in different alphabets, which is an implicit form of Markov imbedding (Fu and Koutras, 1994).

Our strategy relies on the knowledge of two essential parameters: the number of duplicates N and their divergence rate μ. These quantities can be estimated efficiently using the FM-index (Ferragina and Manzini, 2005) as shown in our related work on a prototype mapper based on the concepts developed in this article (Zorita et al., 2020).

The method presented here is general, but it is important to clearly state the assumptions it depends on. First and most importantly, we have ignored insertions and deletions. We assume that the sequencing errors are substitutions only, which makes the method adapted to the Illumina technology, but not to deletion-prone instruments such as the Oxford Nanopore technology. We also assume that insertions and deletions never occur among duplicated sequences. This is obviously incorrect, but our initial tests with real data suggest that this is a minor impediment. Incorporating insertions and deletions would make the theory intractable, so it is presently unclear how to deal with this type of error.

The second assumption is that the candidate set consists of all the genomic locations that have a perfect match for at least one seed, and that all the elements of the candidate set are tested with an exact sequence alignment algorithm. This is possible, but it is important to note that for plant and animal genomes, a single seed may have tens of thousands of hits. Therefore, most mappers impose a limit on the number of alignments per read, opening the possibility that the best hit is seeded but not aligned. It is clear that in this case, the probability that the target is in the candidate set has little to do with the probability that the read is mapped correctly. This is again a minor impediment, since the probability of mapping such reads correctly is low either way. Mappers can have a lower range of confidence score when the candidate set is too large to check every sequence.

The third assumption is that all the duplicate sequences evolve independently of each other and at the same rate. This is again incorrect because duplication events can happen continuously, creating complex ancestry relationships. It is possible to infer the ancestry using tree reconstruction techniques, but it would be challenging to incorporate this information in the present theory. The symbols of the alphabets developed above implicitly assume that the sequences are exchangeable and the complexity of the calculations explodes if it is not the case.

The last assumption is that seeds can match only the target or its duplicates. This does not hold in general because the candidate set usually contains spurious random hits, but we have shown how to deal with this possibility *a posteriori* in section 8.3.

We have not computed off-target probabilities for spaced seeds. In section 3.5 we highlighted the general strategy to compute on-target probabilities with our approach. Traditionally, such on-target probabilities have been computed using finite automata (Buhler et al., 2005; Kucherov et al., 2005), with the caveat that such automata can contain a very large amount of states. There is thus an interest in developing methods to reduce the number of states so that computations can be performed fast (Martin and Noé, 2017). Interestingly, the method presented in section 3.5 generates $2^m + 1$ states (this is the size of the transfer matrix), where m is the number of don't-care positions in the seed model. This is typically lower than the number of states in the automata, but proper benchmarks would be required to know if our method brings real benefits.

Computing the off-target probabilities for spaced seeds can be done with the strategy presented in section 4, but this brings the dimension of the transfer matrix to $4^m + 1$. To give a concrete idea, the seed of PatternHunter (Ma et al., 2002) has 7 don't-care positions, meaning that the transfer matrix would have 16,385 rows and columns. Even though the matrix is sparse, the computation time can be expected to be prohibitive.

Being able to compute seeding probabilities revealed some interesting facts (sections 4.6, 5.7, and 6.8). The first is that the seeding schemes considered here have a worst case scenario for a particular value of μ, the divergence between duplicates. Importantly, the worst value varies between different seeding methods, so it is possible in theory to construct opportunistic seeding strategies that pick the best method for every read, depending on the value of μ. Another interesting fact is that skip seeds can have better performance than exact seeds in the sense that they can yield lower off-seeding probabilities (section 5.7). However, this always comes at the cost of accuracy because skipping nucleotides reduces the probability of on-target seeding.

We also observed that MEM seeds have a significantly higher off-target seeding rate compared to exact seeds and skip seeds (section 6.8). This does not mean that MEM seeding is a bad strategy (it is usually faster than the other methods), but it is good practice to keep an eye on performance and switch methods or even skip the read altogether if the chances of discovering the target are too low. We also showed in section 8.1 that for MEM seeds, the on-target seeding rate is close to the probability that the read is mapped to the true location, which was not the case for exact seeds and skip seeds. In this regard, the present theory is most useful when using MEM seeds.

Regarding the methodology, the Monte Carlo approach of algorithm 1 is relatively straightforward, so one may wonder why the approach with weighted generating functions would be necessary at all for MEM seeds. The only reason is precision. To estimate the frequency of an event by Monte Carlo sampling, this event must occur at least a few times in the simulation. For instance, with 1 million rounds of sampling, frequencies around $1/100,000$ or lower cannot be measured accurately. When one is interested only in frequent events, it is thus a reasonable strategy.

On the other hand, for $N < 20$, the probability that MEM seeding is null or off-target is relatively small, so we need a method that is accurate in this range. Fortunately, the transfer matrix method is fast because the dimension of the matrix $\mathring{M}_N(z)$ is small and the computations are not prohibitive for small values of N.

The proposed methods meet the demand for speed. One needs to compute the probabilities only once for a given value of N and μ (the error rate p is known and constant). For $N > 20$, the iterative method is usually too slow and we need to use Monte Carlo sampling instead. The running time depends on p, on the size of the reads k, and on the desired number of iterations. Since those are constant throughout the sequencing run, the method always takes the same amount of time (around 1-10 seconds for 1,000,000 simulations of reads of size around 100 nucleotides on modern hardware). The values of N and μ can be binned in intervals so that there are only around 100 pairs for a total cost of a few minutes per run. Considering that mappers seem to process at most $10,000$ reads per second per core, the time of mapping a sequencing run of 250 million reads is over 7 h per core, two orders of magnitude larger than the time required to estimate the probabilities of error.

Finally, one may wonder if our approach has any advantage over methods based on machine learning. Such algorithms have already proved useful (Lee et al., 2014) and the rapid progress in the field of deep learning suggests that it is possible to train algorithms to accurately estimate mapping quality. In time, such algorithms may prove faster and/or more robust because they could learn intrinsic biases of the mapping algorithms. Yet, the main benefit of our approach will remain: the combinatorial constructions are a direct access to the nature of the problem. For instance, viewing MEM seeds through the lens of hard and soft masks turns a seemingly intractable process into a relatively simple one (see algorithm 1). The combinatorial stance is that there is value in the models themselves.

In conclusion, we presented a practical solution to the problem of estimating the probability of false positives when using seeding heuristics. This solution is adapted for mapping short reads sequenced with the Illumina technology. Being able to calibrate the seeding heuristic not only allows the user to choose how to balance speed versus accuracy, but also opens new applications. For instance, one can map reads from contaminated samples in pools of closely related genomes (e.g., modern human and Neanderthal) in order to assign the reads to the organism they belong to. In this case, the probabilities of false positives give the right level of confidence in the assignment.

More generally, the analytic combinatorics programme is a very powerful tool to address problems in bioinformatics. Here we have seen how this strategy can be useful even when the generating functions cannot be computed. Using the same ideas, one could calibrate heuristics used in other alignment methods, especially in the expanding field of long-read technologies.

DATA AVAILABILITY STATEMENT

All datasets generated for this study are included in the manuscript/**Supplementary Files**.

AUTHOR CONTRIBUTIONS

GF designed the theory, wrote the C library, and wrote the manuscript. RC and EZ contributed to the theory and edited the manuscript.

ACKNOWLEDGMENTS

We acknowledge the financial support of the Spanish Ministry of Economy, Industry and Competitiveness (Centro de Excelencia Severo Ochoa 2013–2017, Plan Estatal PGC2018-099807-B-I00), of the CERCA Programme/Generalitat de Catalunya, and of the European Research Council (Synergy Grant 609989). RC was supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement 608959. We also acknowledge support of the Spanish Ministry of Economy and Competitiveness (MEIC) to the EMBL partnership.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2020.00572/full#supplementary-material

REFERENCES

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410. doi: 10.1016/S0022-2836(05)80360-2

Arratia, R., and Gordon, L. (1989). Tutorial on large deviations for the binomial distribution. Bull. Math. Biol. 51, 125–131. doi: 10.1016/S0092-8240(89)80052-7

Birol, I., Chu, J., Mohamadi, H., Jackman, S. D., Raghavan, K., Vandervalk, B. P., et al. (2015). Spaced seed data structures for *de novo* assembly. *Int. J. Genomics* 2015:196591. doi: 10.1155/2015/196591

Brejová, B., Brown, D. G., and Vinař, T. (2003). "Vector seeds: an extension to spaced seeds allows substantial improvements in sensitivity and specificity," in *International Workshop on Algorithms in Bioinformatics* (Springer), 39–54.

Břinda, K., Sykulski, M., and Kucherov, G. (2015). Spaced seeds improve k-mer-based metagenomic classification. *Bioinformatics* 31, 3584–3592. doi: 10.1093/bioinformatics/btv419

Buhler, J., Keich, U., and Sun, Y. (2005). Designing seeds for similarity search in genomic DNA. *J. Comput. Syst. Sci.* 70, 342–363. doi: 10.1016/j.jcss.2004.12.003

Burrows, M., and Wheeler, D. (1994). *A Block-Sorting Lossless Data Compression Algorithm*. Digital SRC Research Report. Citeseer.

Califano, A., and Rigoutsos, I. (1993). "Flash: a fast look-up algorithm for string homology," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 353–359.

Chen, Y., Souaiaia, T., and Chen, T. (2009). PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics* 25, 2514–2521. doi: 10.1093/bioinformatics/btp486

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., (2013). Star: ultrafast universal rna-seq aligner. *Bioinformatics* 29, 15–21. doi: 10.1093/bioinformatics/bts635

Faro, S., and Lecroq, T. (2013). The exact online string matching problem: a review of the most recent results. *ACM Comput. Surveys* 45:13. doi: 10.1145/2431211.2431212

Fernandes, F., and Freitas, A. T. (2014). slaMEM: efficient retrieval of maximal exact matches using a sampled LCP array. *Bioinformatics* 30, 464–471. doi: 10.1093/bioinformatics/btt706

Ferragina, P., and Manzini, G. (2000). "Opportunistic data structures with applications," in *Proceedings of 41st Annual Symposium on Foundations of Computer Science,* 390–398.

Ferragina, P., and Manzini, G. (2005). Indexing compressed text. *J. ACM* 52, 552–581. doi: 10.1145/1082036.1082039

Filion, G. J. (2017). Analytic combinatorics for bioinformatics I: seeding methods. *bioRxiv*. 11:205427. doi: 10.1101/205427

Filion, G. J. (2018). Analytic combinatorics for computing seeding probabilities. *Algorithms* 11:3. doi: 10.3390/a11010003

Flajolet, P., and Sedgewick, F. (2009). *Analytic Combinatorics.* Cambridge University Press.

Fu, J. C., and Koutras, M. V. (1994). Distribution theory of runs: a Markov chain approach. *J. Am. Stat. Assoc.* 89, 1050–1058.

Gagie, T., Manzini, G., and Valenzuela, D. (2017). Compressed spaced suffix arrays. *Math. Comput. Sci.* 11, 151–157. doi: 10.1007/s11786-016-0283-z

Healy, J. (2016). "Flak: Ultra-fast fuzzy whole genome alignment," in *Interna- tional Conference on Practical Applications of Computational Biology & Bioinformatics* (Springer), 123–131.

Horton, P., Kielbasa, S. M., and Frith, M. C. (2008). "Dislex: a transformation for discontiguous suffix array construction," in *Proceedings of the Workshop on Knowledge, Language, and Learning in Bioinformatics, KLLBI. Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, 1–11.

Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2012). ART: a next-generation sequencing read simulator. *Bioinformatics* 28, 593–594. doi: 10.1093/bioinformatics/btr708

Jocham, D., Schmiedt, E., Baumgartner, R., and Unsöld, E. (1986). Integral laser-photodynamic treatment of multifocal bladder carcinoma photosensitized by hematoporphyrin derivative. *Eur. Urol.* 12(Suppl. 1), 43–46. doi: 10.1159/000472699

Kachitvichyanukul, V., and Schmeiser, B. (1988). Binomial random variate generation. *Commun. ACM* 31, 216–222.

Khan, Z., Bloom, J. S., Kruglyak, L., and Singh, M. (2009). A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays. *Bioinformatics* 25, 1609–1616. doi: 10.1093/bioinformatics/btp275

Khiste, N., and Ilie, L. (2015). E-MEM: efficient computation of maximal exact matches for very large genomes. *Bioinformatics* 31, 509–514. doi: 10.1093/bioinformatics/btu687

Kiełbasa, S. M., Wan, R., Sato, K., Horton, P., and Frith, M. C. (2011). Adaptive seeds tame genomic sequence comparison. *Genome Res.* 21, 487–493. doi: 10.1101/gr.113985.110

Kucherov, G., Noé, L., and Roytberg, M. (2005). Multiseed lossless filtration. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 2, 51–61. doi: 10.1109/TCBB.2005.12

Kucherov, G., Noé, L., and Roytberg, M. (2006). A unifying framework for seed sensitivity and its application to subset seeds. *J. Bioinform. Comput. Biol.* 4, 553–569. doi: 10.1142/S0219720006001977

Langmead, B., and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nat. Methods* 9, 357–359. doi: 10.1038/nmeth.1923

Lee, W. P., Stromberg, M. P., Ward, A., Stewart, C., Garrison, E. P., and Marth, G. T. (2014). MOSAIK: a hash-based algorithm for accurate next-generation sequencing short-read mapping. *PLoS ONE* 9:e90581. doi: 10.1371/journal.pone.0090581

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *[arXiv preprint] arXiv:1303.3997.*

Li, H., and Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinformatics* 11, 473–483. doi: 10.1093/bib/bbq015

Li, M., Ma, B., Kisman, D., and Tromp, J. (2004). Patternhunter II: highly sensitive and fast homology search. *J. Bioinform. Comput. Biol.* 2, 417–439. doi: 10.1142/S0219720004000661

Li, M., Ma, B., and Zhang, L. (2006). "Superiority and complexity of the spaced seeds," in *Symposium on Discrete Algorithms: Proceedings of the Sev- enteenth Annual ACM-SIAM Symposium on Discrete Algorithm, Vol. 22*, 444–453. doi: 10.1007/978-1-4939-2864-4_803

Lin, H., Zhang, Z., Zhang, M. Q., Ma, B., and Li, M. (2008). ZOOM! Zillions of oligos mapped. *Bioinformatics* 24, 2431–2437. doi: 10.1093/bioinformatics/btn416

Lipman, D. J., and Pearson, W. R. (1985). Rapid and sensitive protein similarity searches. *Science* 227, 1435–1441. doi: 10.1126/science.2983426

Ma, B., Tromp, J., and Li, M. (2002). PatternHunter: faster and more sensitive homology search. *Bioinformatics* 18, 440–445. doi: 10.1093/bioinformatics/18.3.440

Manber, U., and Myers, G. (1993). Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.* 22, 935–948.

Manekar, S. C., and Sathe, S. R. (2018). A benchmark study of k-mer counting methods for high-throughput sequencing. *Gigascience* 7. doi: 10.1093/gigascience/giy125

Martin, D. E. K., and Noé, L. (2017). Faster exact distributions of pattern statistics through sequential elimination of states. *Ann. Inst. Stat. Math.* 69, 231–248. doi: 10.1007/s10463-015-0540-y

Menzel, P., Frellsen, J., Plass, M., Rasmussen, S. H., and Krogh, A. (2013). On the accuracy of short read mapping. *Methods Mol. Biol.* 1038, 39–59. doi: 10.1007/978-1-62703-514-9_3

Nakamura, K., Oshima, T., Morimoto, T., Ikeda, S., Yoshikawa, H., Shiwa, Y., et al. (2011). Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Res.* 39:e90. doi: 10.1093/nar/gkr344

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surv.* 33, 31–88. doi: 10.1145/375360.375365

Needleman, S. B., and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48, 443–453. doi: 10.1016/0022-2836(70)90057-4

Nicodeme, P., Salvy, B., and Flajolet, P. (2002). Motif statistics. *Theor. Comput. Sci.* 287, 593–617. doi: 10.1016/S0304-3975(01)00264-X

Ounit, R., and Lonardi, S. (2016). Higher classification sensitivity of short metagenomic reads with CLARK-S. *Bioinformatics* 32, 3823–3825. doi: 10.1093/bioinformatics/btw542

Régnier, M. A (2000). unified approach to word occurrence probabilities. *Discrete Appl. Math.* 104, 259–280. doi: 10.1016/S0166-218X(00)00195-5

Rumble, S. M., Lacroute, P., Dalca, A. V., Fiume, M., Sidow, A., and Brudno, M. (2009). SHRiMP: accurate mapping of short color-space reads. *PLoS Comput. Biol.* 5:e1000386. doi: 10.1371/journal.pcbi.1000386

Sedgewick, R., and Flajolet, P. (2013) *An Introduction to the Analysis of Algorithms*. Addison-Wesley.

Smith, T. F., and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197. doi: 10.1016/0022-2836(81)90087-5

Sovic, I., Šikic, M., Wilm, A., Fenlon, S. N., Chen, S., and Nagarajan, N. (2016). Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nat. Commun.* 7:11307. doi: 10.1038/ncomms11307

Sun, Y., and Buhler, J. (2005). Designing multiple simultaneous seeds for DNA similarity search. *J. Comput. Biol.* 12, 847–861. doi: 10.1089/cmb.2005.12.847

Sun, Y., and Buhler, J. (2006). Choosing the best heuristic for seeded alignment of DNA sequences. *BMC Bioinformatics* 7:133. doi: 10.1186/1471-2105-7-133

Vyverman, M., Baets, B. D., Fack, V., and Dawyndt, P. (2013). essaMEM: finding maximal exact matches using enhanced sparse suffix arrays. *Bioinformatics* 29, 802–804. doi: 10.1093/bioinformatics/btt042

Waterman, M. S. (1984). General methods of sequence comparison. *Bull. Math. Biol.* 46, 473–500.

Xu, J., Brown J., Li, M., and Ma, B. (2006). Optimizing multiple spaced seeds for homology search. *J. Comput. Biol.* 13, 1355–1368. doi: 10.1089/cmb.2006.13.1355

Zorita, E. V., Cortini, R., and Filion, G. J. (2020). Mapping short reads, faithfully. *BioRxiv*. doi: 10.1101/2020.02.10.942599. [Epub ahead of print].

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.