A Coverage Criterion for Spaced Seeds and its applications to Support Vector Machine String Kernels and k-mer Distances

Laurent Noé [1] and Donald E. K. Martin [2]

[1] LIFL (UMR 8022 Lille 1/CNRS) - Inria Lille, Villeneuve d'Ascq, France laurent.noe@univ-lille1.fr

[2] Department of Statistics, North Carolina State University, Raleigh, NC, USA martin@stat.ncsu.edu

November 16, 2014

Abstract

Spaced seeds have been recently shown to not only detect more alignments, but also to give a more accurate measure of phylogenetic distances (Boden et al., 2013, Horwege et al., 2014, Leimeister et al., 2014), and to provide a lower misclassification rate when used with Support Vector Machines (SVMs) (Onodera and Shibuya, 2013), We confirm by independent experiments these two results, and propose in this article to use a *coverage criterion* (Benson and Mak, 2008, Martin, 2013, Martin and Noé, 2014), to measure the seed efficiency in both cases in order to design better seed patterns. We show first how this *coverage criterion* can be directly measured by a full automaton-based approach. We then illustrate how this criterion performs when compared with two other criteria frequently used, namely the *single-hit* and *multiple-hit criteria*, through correlation coefficients with the correct classification/the true distance. At the end, for alignment-free distances, we propose an extension by adopting the *coverage criterion*, show how it performs, and indicate how it can be efficiently computed.

Keywords. Spaced seed, Spaced k-mer, Gapped k-mer, Coverage sensitivity, Support Vector Machine, String kernel, Alignment-free distance.

1 Introduction

To detect similarities in bio-sequences, in the so called *hit and extend* strategy framework, spaced seeds are now a frequently used technique to define the *hit* (Keich et al., 2004). Several tools have been proposed that

use spaced seeds (Chen et al., 2009, David et al., 2011, Harris, 2007, Homer et al., 2009, Ilie et al., 2013, Kiełbasa et al., 2011, Li et al., 2004, Lin et al., 2008, Zhou et al., 2010), or to design spaced seeds (Buhler et al., 2005, Do Duc et al., 2012, Ilie et al., 2011, Kucherov et al., 2006, Marschall et al., 2012, Nuel, 2011). Work related to spaced seeds also includes the *lossless* filtration problem (Battaglia et al., 2009, Břinda, 2014, Burkhardt and Kärkkäinen, 2002, Egidi and Manzini, 2014a,b, Farach-Colton et al., 2007, Giladi et al., 2010, Kucherov et al., 2005, Nicolas and Rivals, 2008), in the sense that all the alignments of a given set must be detected; the work proposed in this article can be applied to this problem too (section 3.3), but we concentrate on the *lossy* filtration problem in the sense that we suppose that the alignments are associated with a probabilistic model. We also mention a related work on *clump statistics* (Bassino et al., 2008, Marschall et al., 2012, Martin and Coleman, 2011, Régnier et al., 2014, Stefanov et al., 2007) that is close (but not similar), and that can, in some way, be complementary when both of them are considered in a more general framework.

The organization of the article is as follows. Section 2 gives notation and definitions related to spaced seeds. Section 3 defines the *coverage* of spaced seeds and proposes the tools used to measure it. Section 4 shows how *coverage* can be used in two biologically oriented applications : first, when spaced seeds are included within SVM kernels (sub-section 4.1), or when spaced seeds are applied to measure phylogenetic distances (sub-section 4.2). In this last case, we also propose a new distance based on the coverage (sub-section 4.2.2) and the substantial improvement achieved. Section 5 provides, at the end, some concluding remarks.

2 Notation

We suppose here that strings are indexed starting from position number 1. For a given string u, we will use the following notation : $u[i]$ gives the i-th symbol of u, $|u|$ is the length of u, and $|u|_a$ is the number of symbol letters a that u contains. Also, ${}_d(u)$ is the prefix of length d of the string u, and $(u)_d$ is the suffix of length d of the string u. For two strings u and v, $u \cdot v$ is the concatenated string.

Alignments without gaps (*indels*) can be modeled by a succession of *mismatch* or *match* symbols, and thus be represented as a string x in a binary alphabet $\{0, 1\}$. A spaced seed can be represented as a string π, but in a different binary alphabet $\{*, 1\}$: 1 indicates a position on the seed π where a *match* must occur in the alignment x (it is thus called a *must match* symbol), whereas $*$ indicates a position where a *match* or a *mismatch* is allowed (it is thus called a *joker* symbol). The *weight* of a seed π (denoted by w) is defined

as the number of *must match* symbols ($w = |\pi|_1$), whereas the *span/length* of a seed π (denoted by k) is its full length ($k = |\pi|$).

A spaced seed π of length k *hits* an alignment x of length n starting at position i ($i \in [1 \dots n-k+1]$) iff

$$\forall j \in [1 \dots k] \qquad \pi[j] = \mathtt{1} \implies x[j+i-1] = \mathtt{1}$$

The usual requirement for a seed, when used to detect alignments x, is to have *at least one hit* (Keich et al., 2004) in x, the so called *single hit* criterion. Several methods are also based on *multiple hits*, as they require more than one hit to trigger an alignment extension (Burkhardt et al., 1999, David et al., 2011, Rasmussen et al., 2006). In the next section, we extend the way to define criteria based on seed hits by measuring *coverage* provided by these hits.

3 Definition and computation of the seed coverage

3.1 Definition of the coverage

The *coverage* of a seed π on an alignment x is defined by the number of 1's in the alignment x that are covered by *at least* one *must match* symbol of one of the seed's hits (Benson and Mak, 2008, Martin, 2013, Martin and Noé, 2014).

For example, the seed $\pi = \mathtt{11*1}$ has three hits on the string alignment $x = \mathtt{101111001011111}$. The coverage provided by these hits (denoted by • symbols below) is 8.

$\pi_{\ occ_1}$			1	1	*	1										
$\pi_{\ occ_2}$			⋮	⋮		⋮					1	1	*	1		
$\pi_{\ occ_3}$			⋮	⋮		⋮					⋮	1	1	*	1	
x	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	
			•	•		•					•	•	•	•	•	

The coverage concept can be generalized to multiple seed patterns. For example, the set of seeds $\{\pi_1, \pi_2\} = \{\mathtt{11*1}, \mathtt{1*1*1}\}$ has six hits on the string alignment x. The coverage provided by these hits is 11.

$\pi_2\ _{occ_1}$	1	*	1	*	1											
$\pi_1\ _{occ_2}$	⋮		1	1	*	1										
$\pi_2\ _{occ_3}$	⋮		⋮	⋮	⋮	⋮			1	*	1	*	1			
$\pi_1\ _{occ_4}$	⋮		⋮	⋮	⋮	⋮			⋮		1	1	*	1		
$\pi_2\ _{occ_5}$	⋮		⋮	⋮	⋮	⋮			⋮		1	*	1	*	1	
$\pi_1\ _{occ_6}$	⋮		⋮	⋮	⋮	⋮			⋮		⋮	1	1	*	1	
x	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	
	•		•	•	•	•			•		•	•	•	•	•	

3.2 Coverage automaton

Given a seed π or a set of seeds $\{\pi_1, \pi_2, \ldots\}$ along with an input string x, the aim of the automaton is to compute the coverage of $\pi_1, \pi_2, \ldots$ on x, as defined in section 3.1. To fully compute the coverage, a necessary and sufficient task, typically devoted to an automaton, is to update the coverage each time we concatenate a new symbol to the right of x. For example, for the set of seeds $\{\pi_1, \pi_2\} = \{\texttt{11}*\texttt{1}, \texttt{1}*\texttt{1}*\texttt{1}\}$ and the string $x = \texttt{1011110011110}$, we desire to determine the set of newly covered positions (denoted by two $\circ$ symbols below) after reading the new symbol 1 to form the extended string $x' = x \cdot 1$, together with their count to update the coverage. We will call this (+2) value the *coverage increment*.

π_2 occ_1	1	*	1	*	1									
π_1 occ_2	⋮		1	1	*	1								
π_1 occ_3	⋮		⋮	⋮	⋮	⋮			1	1	*	1		
x	1	0	1	1	1	1	0	0	1	1	1	1	0	
	•		•	•	•	•			•	•		•		
						↓								
π_2 occ_1	1	*	1	*	1									
π_1 occ_2	⋮		1	1	*	1								
π_1 occ_3	⋮		⋮	⋮	⋮	⋮			1	1	*	1		
π_2 occ_4	⋮		⋮	⋮	⋮	⋮			⋮	1	*	1	*	1
π_1 occ_5	⋮		⋮	⋮	⋮	⋮			⋮	⋮	1	1	*	1
x'	1	0	1	1	1	1	0	0	1	1	1	1	0	1
	•		•	•	•	•			•	•	○	•		○

For a set of seeds $\{\pi_1, \pi_2, \ldots\}$ with $k = max_i(|\pi_i|)$, first notice that a suffix of x of length (at most) $k-1$ is sufficient to know which *proper prefixes* of one of the seeds can lead to a new hit : we will call q this suffix. Moreover, to update the coverage increment, we need to know which 1 symbols inside q have already been covered by previous hits of one of the seeds : this can be done with a binary word c of length $|q|$ associated with q. States of the automaton are thus defined accordingly by a pair $\langle {}^{q}_{c} \rangle$.

For example, for the set of seeds $\{\pi_1, \pi_2\} = \{\texttt{11}*\texttt{1}, \texttt{1}*\texttt{1}*\texttt{1}\}$, the state reached when reading the first string alignment $x = \texttt{1011110011110}$ (used in the previous example) is represented by the pair $\langle {}^{q}_{c} \rangle = \langle {}^{1110}_{\bullet\,\bullet} \rangle$ and the transition to $x' = x \cdot 1$ can be computed accordingly with the new hits of π_1 and π_2

π_2 occ_4	...	1	*	1	*	1
π_1 occ_5	...		1	1	*	1
$q \to q'$	...	1	1	1	0	1
		•	○	•		○

Figure 1: Minimized Mealy coverage automaton (count on transitions) for the seeds $\{\pi_1, \pi_2\} = \{11 * 1, 1 * 1 * 1\}$

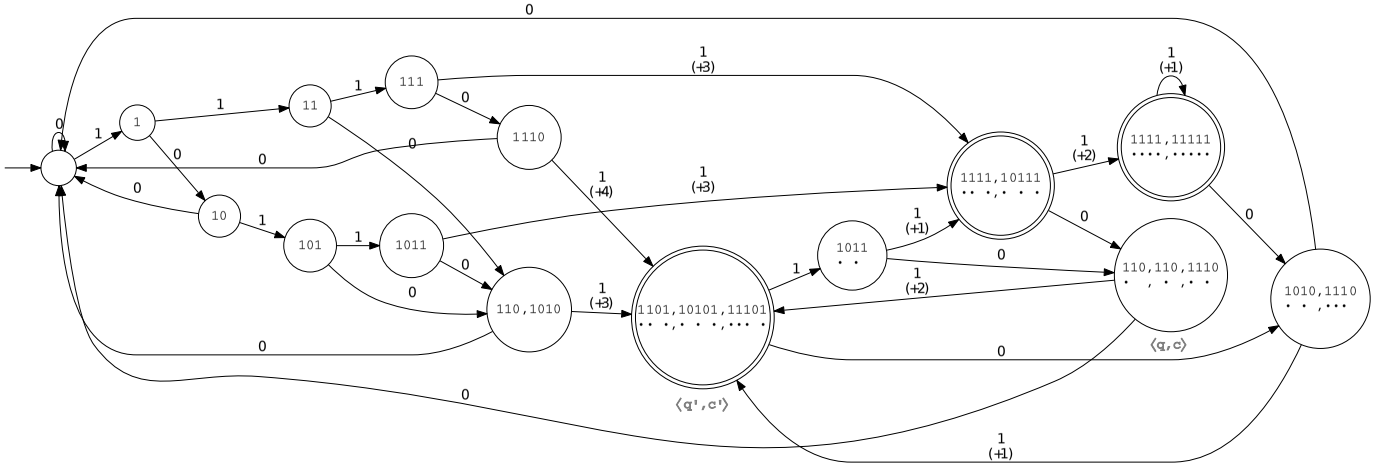
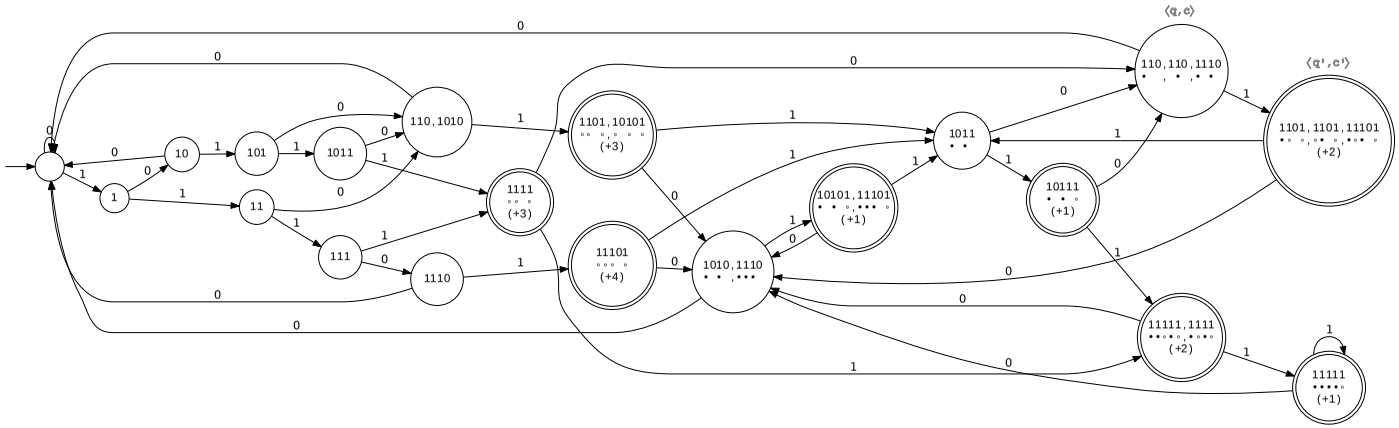


Figure 2: Minimized Moore coverage automaton (count on states) for the seeds $\{\pi_1, \pi_2\} = \{11 * 1, 1 * 1 * 1\}$



The new state may be represented by the pair $\langle {}^{q'}_{c'} \rangle = \langle {}^{1101}_{\bullet\bullet\ \bullet} \rangle$ with $|q'| \leq k-1$. Note that q' can even be reduced to a smaller suffix, because no *proper prefix* of π_1 or π_2 can start with $q' = \texttt{1101}$, but a prefix of π_2 can match the first proper suffix of q', namely $\texttt{101}$, to initiate a hit. Thus $\langle {}^{q'}_{c'} \rangle = \langle {}^{1101}_{\bullet\bullet\ \bullet} \rangle \equiv \langle {}^{101}_{\bullet\ \bullet} \rangle$: this reduction can be done easily using the Fail function of Aho-Corasick algorithm (Aho and Corasick, 1975) which is applied in classical seed automata (Buhler et al., 2005, Kucherov et al., 2007), as well as coverage automata (Benson and Mak, 2008, Martin and Noé, 2014). We will suppose that we always apply this reduction on all the states $\langle {}^{q}_{c} \rangle$.

From the point of view of the automaton definition, two finite state machines are possible : *Mealy* or *Moore*. Accordingly, the automaton must provide the *coverage increment*, either on each transition (for the *Mealy* automaton), or on each state (for the *Moore* automaton). For example, on the set of seeds $\{\pi_1, \pi_2\} = \{\texttt{11}*\texttt{1}, \texttt{1}*\texttt{1}*\texttt{1}\}$, these two representations are illustrated on Figures 1 and 2 : due to size, we present here the minimal version for both automata by merging equivalent states. For readability, when some hits occur, we have represented the states $\langle {}^{q}_{c} \rangle$ with their full length matching symbols of length up to k and not $k-1$ (see for example $\langle {}^{q}_{c} \rangle \equiv \langle {}^{1101}_{\bullet\bullet\ \bullet} \rangle$ and $\langle {}^{q'}_{c'} \rangle \equiv \langle {}^{101}_{\bullet\ \bullet} \rangle$ on the Figures 1 and 2).

The *Mealy* automaton is obviously more compact when considering the number of states. On the other hand, it requires one to store an additional value per transition (and also needs more specific algorithms : for example, the Hopcroft minimization algorithm (Hopcroft, 1971) must be adapted to the Mealy case).

Each representation has been independently implemented by one of the authors : the one based on *count on transition* (*Mealy*) is implemented in Matlab (see Martin, 2013, Martin and Noé, 2014), and code has been also tested on Octave (Octave community, 2014), whereas the other, mainly for compatibility issues, is based on *count on states* (*Moore*), and is generalized for subset seeds (slight extension of spaced seeds) with multiple seeds in mind (Kucherov et al., 2007). The "Mealy" Matlab code is available upon request from the second author, and the "Moore" code is included in the C++ Iedera program (Kucherov et al., 2014) starting from development version 1.06 α7.

Several minimizations of the states (considering both q and c) can be considered *during* the construction of these automata, but the details are out of scope of this article (see Kucherov et al., 2007, Martin and Noé, 2014). In practice, we use at least two methods to detect coverage strings c that are *equivalent*, together with the optimisation of Kucherov et al. (2007) on strings q to save some memory space **before** completing the full automaton. Note that this last automaton, once entirely built, can always be fully reduced to its minimal form, for example by applying the classical Hopcroft minimization algorithm (Hopcroft, 1971).

Independently, we also mention that it seems difficult, for this special *coverage* problem, to find an

equivalent classical *regular expression* to help build the automata. Even classical tools (such as `grep`) have for example equivalent parameters to simulate a *single* or *multiple* hit, but no parameter is provided for this *coverage* problem.

3.3 Computation

Given a generative model $\mathcal{X}$ for the string x, it is possible to compute the distribution of the coverage values according to a Markov process (Martin, 2013, Martin and Noé, 2014) or any model that can be represented by a non-deterministic probabilistic automaton (Kucherov et al., 2006, Marschall et al., 2012, Martin and Noé, 2014, Nuel, 2008). We don't consider directly in this article this computation, as the model used in our tests is pure Bernoulli : the computation can thus be performed directly with a simple dynamic programming algorithm on the coverage automaton of section 3.2. We refer to the aforementioned articles for more details on more complex probabilistic models.

Independently, we also mention that the work proposed here is applied on the *lossy seed framework*, in the sense that we consider a *probability* to *hit* (or *cover*) an alignment sequence x generated by a model $\mathcal{X}$. However, this work is not strictly limited to probabilities, and can be easily extended, for example to the *lossless seed framework*. In that case, the set of alignments is fixed, for example by giving a fixed length together with a fixed number of errors : the problem is then to always *hit* (or *cover*) any of the alignments on this set (so without *loss*). This last computation can be done easily, simply by replacing the *semi-ring* used for probabilities by a less conventional *tropical* semi-ring (Mohri, 2009, Pin, 1998, Simon, 1988) used for match/mismatch scores or mismatch costs. Note also that the simple fact of counting the number of alignments, in alignment classes that have a given percentage of identity (as done in Benson and Mak, 2008), or a given coverage for a set of seeds, or any combination of these elements, is also possible, by use of a *counting* semi-ring adapted for this task (Huang, 2006).

4 Experiments

In this section, we consider two *biological sequence* oriented applications that have recently been proposed to use spaced seeds : SVM classifiers based on spaced string kernels (Onodera and Shibuya, 2013), and alignment-free distance estimators using spaced k-mers (Boden et al., 2013, Horwege et al., 2014, Leimeister et al., 2014). We show that the coverage sensitivity can be used in both cases to improve the estimators, and thus also be applied to the selection of the best seed patterns on such domains.

Additional data and results, together with scripts used for this section can be found at `http://bioinfo.lifl.fr/yass/iedera_coverage/`.

4.1 Coverage sensitivity and spaced seed string kernels

String kernels (Lodhi et al., 2002) are a classical model used for text classification with SVM. They have frequently been applied to biological sequence classification, as k-spectrum kernels (Leslie et al., 2002), mismatch k-spectrum kernels (Leslie et al., 2004), string alignment kernels (Saigo et al., 2004), profile-based string kernels (Kuang et al., 2005) to cite a few examples.

k-spectrum kernels and its derivatives are mostly used with *contiguous seeds* : surprisingly, no *spaced seeds* were designed to comply with this approach. However, it has been experimentally shown by Onodera and Shibuya (2013), and during the submission of this work by Ghandi et al. (2014a,b) that spaced seeds help decrease the zero/one misclassification rate in practice, even for the simplest kernels. The main reason of this lack might be the intrinsic difficulty to find a *correct estimation criterion* for spaced seed patterns, but on the other hand, not so much effort has been made to increase the diversity of criteria used. Most of the proposed algorithms to estimate spaced seed sensitivity concentrate on the **single-hit criterion** (*"at least one hit for a seed/set of seeds"*). This criterion makes sense for classical "hit and extend" alignment methods used in bioinformatics, but seems to be too restrictive for spectrum kernels that are supposed to filter the information content based of "several concordant clues".

The **multi-hit criterion** (*"at least t hits for a seed/set of seeds"*) seems at first more appropriate for this task, but again has never been tried in this field of research. One possible drawback is that it does not distinguish highly overlapped hits of seeds from disjoint ones. Finally, and for the latter reason, we also decided to apply the new **coverage criterion** (*"at least t covered* `1`*-symbols in the alignment, each covered by at least one* `1`*-symbol of a seed hit "*) in comparison with the two others.

In the two following subsections, we try to correlate these three criteria with SVM zero/one misclassification rate.

4.1.1 SVM-Benchmark and Protocol

The benchmark used for this test consists of 2208 families extracted from the non-coding RNA database RFAM v11.0 (Burge et al., 2012). It represents up to 65908 sequences per family.

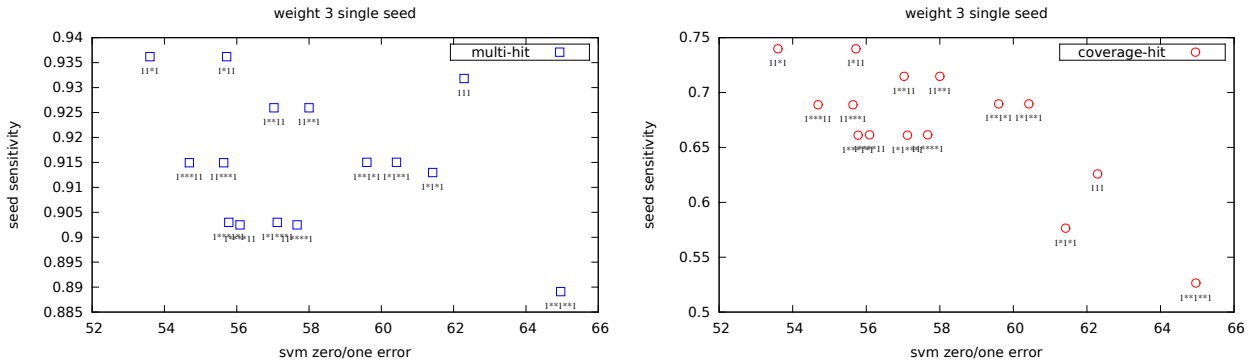
We decided to split each family by randomly picking 50% of its sequences for the SVM learning process and keeping the 50% remaining for the classifier to measure the zero/one misclassification rate. We use

the $SVM^{multiclass}$ (Joachims, 2002) package Version 2.20 (date 14.08.2008) from `http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html` with the linear kernel. In each case, single or double seeds of weight 3 and span from 3 to 7 were used as a k-spectrum string kernel.

4.1.2 Seed sensitivity

In parallel, for each single or double seed, we compute its "sensitivity", either using the single hit criterion, the multi-hit criterion, or the coverage criterion. Note that, for the two last criteria, we have the possibility to change the *threshold* t required to consider a success. We arbitrarily choose to measure these seeds on an i.i.d. alignment model of length 32 (the probability to have a 1-symbol in the alignment has been fixed at 0.7) although experiments show that this does not have much influence on the final results [data not shown].

Figure 3: *zero/one misclassification rate* vs *theoretical sensitivity*



Examples of comparative plots are given in Figure 3 for multi-hit and coverage-hit : a slight correlation can be seen at first sight. But we can also see that some seeds with repetitive and highly correlated patterns (e.g. `1*1*1`), usually bad in theory, are in practice more efficient for the SVM-classifier.

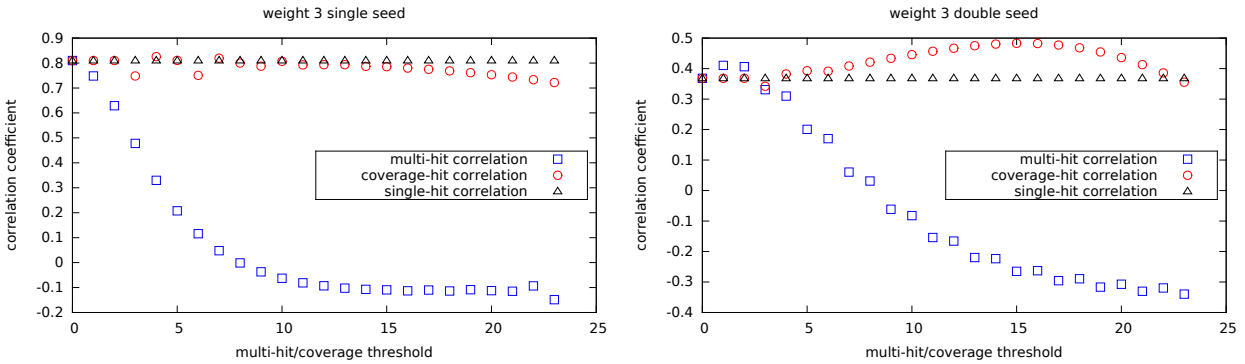
4.1.3 Correlation between zero/one misclassification and the three criteria

Finally, to determine if one of the three estimators was better suited to correlate with this SVM classifier task, we computed the *sample Pearson correlation coefficient* for each of the three criteria, for each set of seeds : this gives the best correlation between the theoretical seed sensitivity estimated by one of the three estimators with the experimentally measured sensitivity of the SVM classifier of each set of seeds.

For both the multiple hit and coverage criteria, we allowed the threshold parameter t for seed sensitivity to vary (x-axis). These results are illustrated in Figure 4 for single and double seeds.

Surprisingly, correlation results for the **multi-hit criterion** are not good when the number of hits

Figure 4: Correlation coefficient between *zero/one misclassification* rate and *theoretical sensitivity*



required is too large. This must be taken into account when using this criterion because the multi-hit criterion gives correct results for double seeds when the number of hits is for example at 2 .

The **single-hit criterion** gives good results for each set. However combining single and double seeds into one set, and doing the same experiment makes it the worse of the three estimators [data not shown]. A carefully chosen value for the **coverage criterion** (here between 14 and 16) helps to reach the highest correlation of the three for double seeds. On single seeds, this is difficult to conclude, due to the few seeds of weight 3 that have been tested. Note that we also tried the same experiment for seeds of weight 4 but the dimension used here (4^4) makes the classifier more random without a preselection of dimensions [data not shown].

We can first notice that the correlation of the single-hit criterion is more stable than the correlation of the coverage criterion that varies more for lower thresholds. It also seems that the optimal coverage threshold is at some point a surprisingly quite regular and convex function that might be estimated when enough data is available.

4.2 Coverage sensitivity and alignment-free distance for sequence comparison

Estimating alignment-free distance is a common method used for sequence comparison in multiple alignment tools guided tree estimation (Edgar, 2004) and related phylogenetic tree estimation (Liu et al., 2008, Qi et al., 2004). Several distances are based on fixed size k-mers (Simsa et al., 2009, Vinga and Almeida, 2003) with possible mismatches allowed (Apostolico et al., 2014), or with variable length k-mers : local decoding (Didier et al., 2012), irredundant common subwords (Comin and Verzotto, 2012), etc. They are applied on assembled genomes (Chor et al., 2009, Haubold et al., 2005), protein classification (Stropea and Moriyama, 2007), and even on unassembled genomic data to estimate phylogenies (Maurer-Stroh et al., 2013). We refer to a recent

special issue on alignment-free methods for more details (Vinga, 2014).

Interestingly, it's only in the last year that the use of spaced seeds has been proposed (Boden et al., 2013, Horwege et al., 2014, Leimeister et al., 2014), with recent applications for specific *Next Generation Sequencing* tasks, such as multi-clonal clusterization (Giraud et al., 2014). Here again, the lack of seed criteria used in the literature didn't help the selection of good seeds for these tasks.

In subsection 4.2.1, we recall that the "classical" distance can be estimated by multi-hit sensitivity computation which helps in selecting good spaced seeds.

In subsections 4.2.2 and 4.2.3, we also show that coverage sensitivity can be used in a more elaborate distance : this distance can be computed using the *Longest Increasing Subsequence (LIS)* of the positions of the common hits between gapped k-mers. As LIS can be computed in $t \cdot \log(t)$ time, where t is the number of hits, it is thus a reasonable estimator in practice.

4.2.1 Multi-hit experimental support

One common method used to estimate alignment-free distances is based on k-mer frequency : 4^k counts can be first made and used as simple *Feature Frequency Profiles* (where counts are normalized to relative frequencies for any of the 4^k k-mers), or more elaborate *Composition Vectors* (where normalization is done with the help of a background model). Distances can then be estimated by several models (Vinga and Almeida, 2003) to provide phylogenetic applications with an initial distance matrix. As some of these phylogenetic methods, as *Unweighted Pair Group Method with Arithmetic Mean* (UPGMA) (Michener and Sokal, 1957), start by considering small distances, it's important to have the best estimator here, and keep track of common k-mers (or spaced k-mers) and their common locations in the two sequences. One estimator that can help in that task is the number of seed hits obtained : we will call it the *multi-hit value*.

For our experiment, we use a set of seeds (627 seeds of weight 3 or 4, span up to 7, single seed or double seed patterns), a percentage of identity varying from 20% to 100% by steps of 5% each time, and we generate (for each percentage of identity) 1000 alignments of length 32. We then measure the *multi-hit value* of each alignment and compare it to the true alignment distance.

It can be shown first (Figure 5 x-axis only) that the correlation coefficient is high (> 0.9 for seeds of weight 3, less otherwise). Provided that we expect to pay a little additional cost, it is possible to improve this result, as shown in the next section.

4.2.2 Coverage experimental support

The distance we propose to measure is based on the number of covered `1`-symbols in the alignment, each covered by at least one `1`-symbol of a seed hit : we will call it the *coverage value*. To show how this distance better estimates the true distance (we assume here that the Hamming distance is the true distance), we are repeating the same experiment with both the *multi-hit value* and the *coverage value* on the set.

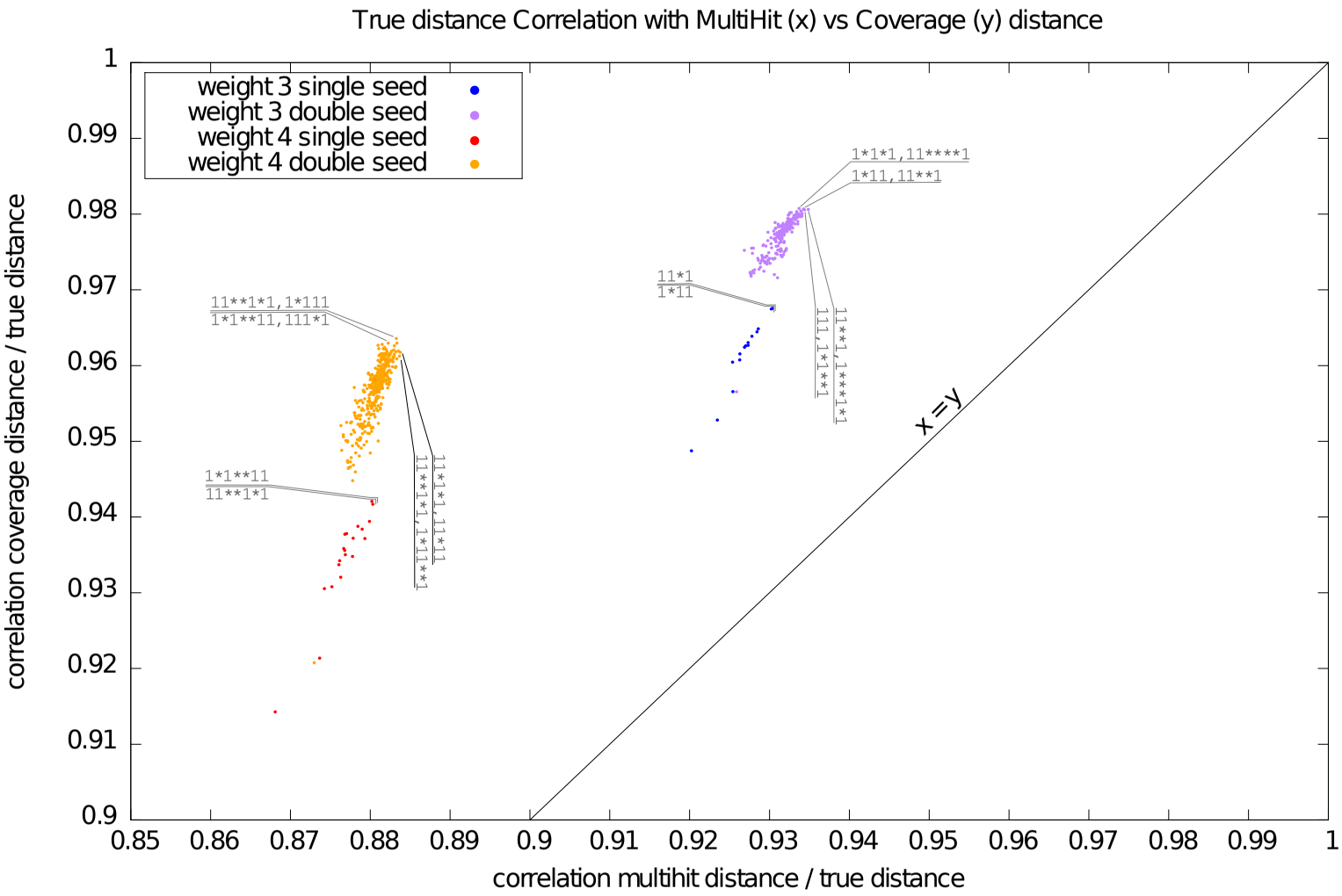
We use the same protocol here : the same set of seeds (627 seeds of weight 3 or 4, span up to 7, single seed or double seed patterns), the same percentage of identity varying from 20% to 100% by steps of 5% each time, and generating for each percentage of identity the same 1000 alignments of length 32 each time, we measure the *multi-hit* **and** *the coverage values* for each simulated alignment. Then, we compared the *correlation coefficient* for each of these two measures with the true percentage of identity used to simulate the alignment.

The *correlation coefficient* for all the seeds was 0.88 for the *multi-hit value* and 0.96 for the *coverage value*. We tried to refine this first experiment by separately measuring single seed patterns and double seed patterns and running the same test. For single seed patterns, the correlation was 0.89 and 0.94 respectively, whereas for multiple seed patterns, it was 0.89 and 0.96 respectively. We also tried to measure this correlation for each of the 627 seeds : Figure 5 plots these two correlations (pair of coordinates).

Note first that, as all the points for this plot are on the *left-upper* region, the true distance is better estimated by the *coverage value* than by the *multi-hit value*. We can also notice that double seeds outperform single seeds in both cases, so that multiple seed patterns can help in estimating the distance more accurately than single seed patterns : the gain is even better for the *coverage value* than for the *multi-hit value*.

From the point of view of the seed weight and the number of seed patterns used, we can see that using *two patterns of weight 4* gives the same correlation coefficient as using *one single pattern of weight 3*, **but only for the coverage value, not for the multi-hit value** : this encouraging result may help defend the idea that more patterns of larger weight will help measure a correct distance. Note that this conclusion is quite similar to the one provided ten years ago for detecting alignments (Li et al., 2004), which was recently and independently observed in Horwege et al. (2014), Leimeister et al. (2014), but here, as the distance estimation problem is quite different from alignment detection, the seeds designed will probably be completely different from those previously seen.

Figure 5:


True distance Correlation with MultiHit (x) vs Coverage (y) distance
weight 3 single seed
weight 3 double seed
weight 4 single seed
weight 4 double seed
1*1*1,11****1
1*11,11**1
11*1
1*11
11**1,1***1*1
111,1*1**1
11**1*1,1*111
1*1**11,111*1
11*1*1,11*11
11**1*1,1*11**1
1*1**11
11**1*1
x=y
correlation coverage distance / true distance
correlation multihit distance / true distance
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
0.85
0.86
0.87
0.88
0.89
0.9
0.91
0.92
0.93
0.94
0.95
0.96
0.97
0.98
0.99
1

From the point of view of the seed patterns, we can see in Figure 5 that, for single seeds, selection done for both *values* gives the same optimal seed pattern `11*1` (or its mirror) for weight 3, and the same optimal seed pattern `1*1**11` (or its mirror) for weight 4. The choice for the optimal double seed patterns differs between the *multi-hit* or *coverage values*, and this difference is even more marked for seeds of weight 4.

However, computing the coverage is more difficult than simply counting common k-mers. We justify in the next part that, given two easily measurable assumptions on the sequences and the k-mer weight, this task can be done efficiently.

4.2.3 Coverage algorithmic point of view

In this part, we briefly describe how coverage can be computed efficiently. Given two sequences s_1 and s_2 of equivalent length, we want to search for the *spaced* k-mers that are common to s_1 and s_2. But, more than establishing a frequency profile for these common k-mer *codes*, the main idea is here to find a set of common k-mers that have the same order of position occurrences on s_1 and s_2. To do so, one solution is to keep occurrences of any of the 4^k possible k-mers in a *reverse list of positions* (given one k-mer code, we have two lists of positions where this k-mer occurs, on s_1 or respectively on s_2). Keeping the common k-mers of both s_1 and s_2, sorting their list of pairs of occurrence positions according to the positions of one of the two sequences (for example positions along s_1), then applying a LIS (or a windowed LIS if the two sequences are not of similar lengths) on s_2, provided that spurious k-mers (those occurring randomly) are not frequent, will give a better approximation for the number of true hits, and thus can be used to compute the coverage.

Note first that the LIS can be computed in $t \cdot \log(t)$ time (Schensted, 1961) where t is the number of *hits* (e.g. pairs of positions for a common k-mer) : this value t, provided that k is well chosen to correctly filter *spurious k-mers* and there is no composition bias on both sequences, must be either close to $|s_1|$ and $|s_2|$ if the s_1 and s_2 sequences are similar (and without self-repetitive bias/low complexity regions), or reasonably low if the sequences are non-similar, but can be otherwise high for low complexity/self-repeating/redundant regions that similarity search tools usually want to avoid.

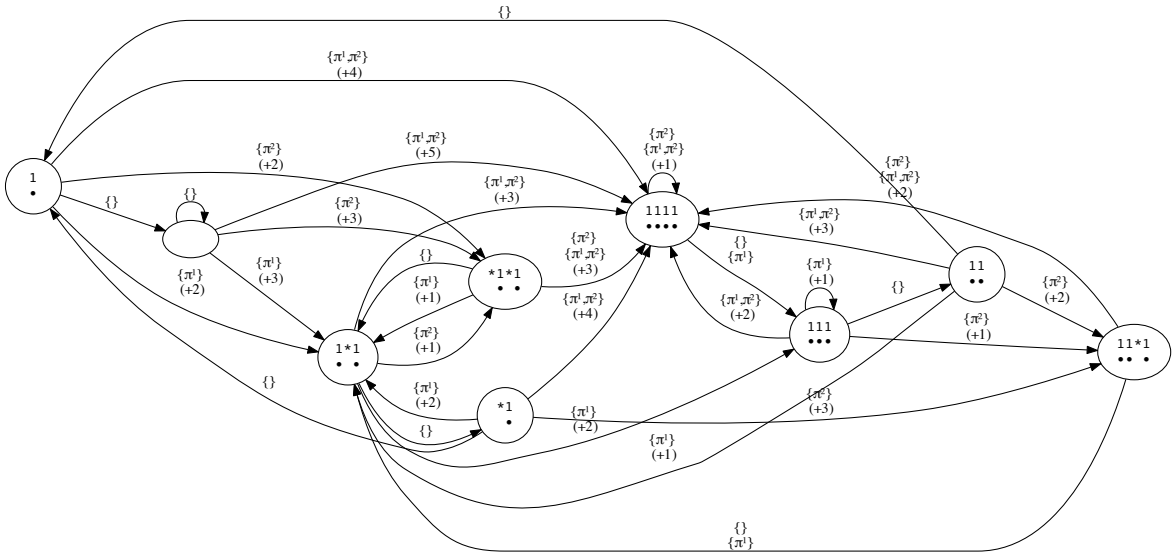
Note also that, once the common and ordered hits are collected by the LIS procedure, it is possible to compute the coverage using :

- either a masking process using shift-or for collecting the coverage symbols, and then computing the coverage increment (which implies an additional CPU cost if no *population count instruction* is available),
- or an automaton (an example is provided in Figure 6 for the hits of the seeds $\{\pi_1, \pi_2\} = \{\texttt{11}*\texttt{1}, \texttt{1}*\texttt{1}*\texttt{1}\}$)

that keeps the last overlapping suffix of the previously encountered hits for any of the seeds. This automaton has an alphabet of size $2^{\#\text{seeds}}$ since we record whether or not there is a seed hit for each seed. Otherwise, a very similar definition to the coverage automaton holds. Once this automaton is built, it is possible to compute the coverage increment in constant time.

In both cases, gaps (*indels*) must be taken into consideration because they break, from a *dot-plot* point of view, *diagonals*, thus reinitializing the automaton or the coverage mask.

Figure 6: Mealy coverage increment automaton for hits of the seeds $\{\pi_1, \pi_2\} = \{\texttt{11}*\texttt{1}, \texttt{1}*\texttt{1}*\texttt{1}\}$



5 Concluding Remarks

We have presented how the *coverage criterion* (Benson and Mak, 2008, Martin, 2013) can help in measuring the seed efficiency in two recent problems : a classifier based on spaced k-mers (Onodera and Shibuya, 2013), and a k-mer alignment-free distance estimation (Boden et al., 2013, Horwege et al., 2014, Leimeister et al., 2014). We have also shown how to extend the second one to be even more sensitive.

The Moore (or Mealy) automaton obtained to measure the coverage criterion is by itself of interest for several reasons : its size seems to be bounded by $polynom(w, r) \times 3^r$ even if the bound obtained now is rather limited and exponential (see the Appendix).

For example the *coverage automaton* size for the PatternHunter 1 seed `111*1**1*1**11*111` is :

Moore `iedera` development version 1.06 α7		Mealy Matlab code + gap-system FR	
current size 4312 states	minimized 4260 states	current size 4215 states	minimized 3782 states

where the *current sizes* for Moore and Mealy automata are respectively obtained by the Iedera tool (Kucherov et al., 2014, version 1.06 α7), or by the Matlab code (Martin and Noé, 2014) before minimization by the gap-system FR package (Bartholdi, 2012). These sizes can be compared with those of the mere *multi-hit automaton* :

Moore `iedera` development version 1.06 α7		Mealy Matlab code + gap-system FR	
current size 322 states	minimized 322 states	current size 281 states	minimized 278 states

Although the coverage automaton is more than ten times larger than the equivalent multi-hit automaton, it is still usable for dynamic programming computation.

This is even true for **multiple spaced seeds**. For example the *coverage automaton* size for the PatternHunter 2 seeds of weight 11 : `111*1**1*1**11*111, 1111**11**1*1****1*11, 11*1****11***1*1*1111, 111*111*1***1111` (called *first four* in Li et al. (2004)) is :

Moore `iedera` development version 1.06 α7		Mealy Matlab code + gap-system FR	
current size 154412 states	minimized 143736 states	current size *not available*	minimized 127049 states

to be compared again with the mere *multi-hit automaton* current size (and its minimal size) :

Moore `iedera` development version 1.06 α7		Mealy Matlab code + gap-system FR	
current size 5119 states	minimized 4963 states	current size *not available*	minimized 4183 states

Although more than 20 times larger than the equivalent multi-hit automaton, the coverage automaton for multiple seeds is again still usable for dynamic programming computation.

It would be also interesting (but out of the scope of this article) to consider SVM kernels or k-mer distances with *subset seed* (Frith and Noé, 2014, Gambin et al., 2011, Kucherov et al., 2007, Yang and Zhang, 2008) or more general *vector seed* (Brejová et al., 2005) techniques. Several string kernels, such as the *mismatch string kernel* (Leslie et al., 2004), use this general concept, but generate full neighborhoods (all the words at a given distance from a given k-mer). Moreover *optimal resolution* [best seed weight] (Simsa

et al., 2009) remains an open problem for spaced seeds in both SVM kernels or k-mer distance problems. Note also, if one wants to avoid this *optimal resolution* question, *seed design* and *increasing weight* can be combined (as done in Csűrös, 2004, Kiełbasa et al., 2011), but may not be always directly compatible with the aforementioned cited works on variable k-mers.

A last idea to explore is also to merge the definition of clumps (Bassino et al., 2008, Marschall et al., 2012, Martin and Coleman, 2011, Régnier et al., 2014, Stefanov et al., 2007) with coverage, for example by giving *more significance* (than a linear weight function) to coverage provided by clumps of hits than coverage provided by isolated hits.

Acknowledgements

D.E.K. Martin was supported in this research by the National Science Foundation under Grant DMS-1107084. L. Noé was supported by a CNRS Mastodons grant, and benefited from a half-time course buyout from the French Institute for Research in Computer Science and Automation (Inria).

Author Disclosure Statement

No competing financial interests exist.

References

Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975. doi: 10.1145/360825.360855. 3.2

Alberto Apostolico, Concettina Guerra, and Cinzia Pizzi. Alignment free sequence similarity with bounded Hamming distance. In *Proceedings of the Data Compression Conference (DCC)*, 2014. doi: 10.1109/DCC.2014.57. 4.2

Laurent Bartholdi. Functionally recursive groups. http://www.gap-system.org/Manuals/pkg/fr-2.1.1/doc/chap0.html, 2012. 5

Frédérique Bassino, Julien Clément, Julien Fayolle, and Pierre Nicodème. Constructions for clumps statistics. *Discrete Mathematics and Theoretical Computer Science*, AI:179–194, 2008. 1, 5

Giovanni Battaglia, Davide Cangelosi, Roberto Grossi, and Nadia Pisanti. Masking patterns in sequences: A new class of motif discovery with don't cares. *Theoretical Computer Science*, 410(43):4327–4340, 2009. doi: 10.1016/j.tcs.2009.07.014. 1

Gary Benson and Denise Y.F. Mak. Exact distribution of a spaced seed statistic for DNA homology detection. In *Proceedings of the International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 5280 of *LNCS*, pages 282–293, 2008. doi: 10.1007/978-3-540-89097-3_27. (document), 3.1, 3.2, 3.3, 5, A.2

Marcus Boden, Martin Schöneich, Sebastian Horwege, Sebastian Lindner, Chris Leimeister, and Burkhard Morgenstern. Alignment-free sequence comparison with spaced k-mers. In *Proceedings of the German Conference on Bioinformatics (GCB)*, volume 34 of *OpenAccess Series in Informatics (OASIcs)*, pages 24–34, 2013. doi: 10.4230/OASIcs.GCB.2013.24. (document), 4, 4.2, 5

Broňa Brejová, Daniel G. Brown, and Tomáš Vinař. Vector seeds: An extension to spaced seeds. *Journal of Computer and System Sciences*, 70(3):364–380, 2005. doi: 10.1016/j.jcss.2004.12.008. 5

Karel Břinda. Languages of lossless seeds. In *Proceedings of the International Conference on Automata and Formal Languages (AFL)*, volume 151, pages 139–150, 2014. doi: 10.4204/EPTCS.151.9. 1

Jeremy Buhler, Uri Keich, and Yanni Sun. Designing seeds for similarity search in genomic DNA. *Journal of Computer and System Sciences*, 70(3):342–363, 2005. doi: 10.1016/j.jcss.2004.12.003. 1, 3.2, A.1

Miklós Csűrös. Performing local similarity searches with variable length seeds. In *Proceedings of the 15th Annual Combinatorial Pattern Matching Symposium (CPM)*, volume 3109 of *LNCS*, pages 373–387, 2004. doi: 10.1007/978-3-540-27801-6_28. 5

Matei David, Misko Dzamba, Dan Lister, Lucian Ilie, and Michael Brudno. SHRiMP2: Sensitive yet practical short read mapping. *Bioinformatics*, 27(7):1011–1012, 2011. doi: 10.1093/bioinformatics/btr046. 1, 2

Gilles Didier, Eduardo Corel, Ivan Laprevotte, Alex Grossmann, and Claudine Landès-Devauchelle. Variable length local decoding and alignment-free sequence comparison. *Theoretical Computer Science*, 462:1–11, 2012. doi: 10.1016/j.tcs.2012.08.005. 4.2

Dong Do Duc, Huy Q. Dinh, Thanh Hai Dang, Kris Laukens, and Xuan Huan Hoang. AcoSeeD: An ant colony optimization for finding optimal spaced seeds in biological sequence search. In *Proceedings of the 8th International Conference on Swarm Intelligence (ANTS)*, volume 7461 of *LNCS*, pages 204–211, 2012. doi: 10.1007/978-3-642-32650-9_19. 1

Robert C. Edgar. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004. doi: 10.1093/nar/gkh340. 4.2

Lavinia Egidi and Giovanni Manzini. Spaced seeds design using perfect rulers. *Fundamenta Informaticae*, 131(2): 187–203, 2014a. doi: 10.3233/FI-2014-1009. 1

Lavinia Egidi and Giovanni Manzini. Design and analysis of periodic multiple seeds. *Theoretical Computer Science*, 522:62–76, 2014b. doi: 10.1016/j.tcs.2013.12.007. 1

Martin Farach-Colton, Gad M. Landau, Süleyman Cenk Sahinalp, and Dekel Tsur. Optimal spaced seeds for faster approximate string matching. *Journal of Computer and System Sciences*, 73(7):1035–1044, 2007. doi: 10.1016/j.jcss.2007.03.007. 1

Martin C. Frith and Laurent Noé. Improved search heuristics find 20 000 new alignments between human and mouse genomes. *Nucleic Acids Research*, 42(7):e59, 2014. doi: 10.1093/nar/gku104. 5

Anna Gambin, Sławomir Lasota, Michał Startek, Macieij Sykulski, Laurent Noé, and Gregory Kucherov. Subset seed extension to Protein BLAST. In *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms*, pages 149–158. SciTePress Digital Library, 2011. doi: 10.5220/0003147601490158. 5

Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A. Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Computational Biology*, 10(7):e1003711, July 2014a. doi: 10.1371/journal.pcbi.1003711. 4.1

Mahmoud Ghandi, Morteza Mohammad-Noori, and Michael A. Beer. Robust k-mer frequency estimation using gapped k-mers. *Journal of Mathematical Biology*, 69(2):469–500, August 2014b. doi: 10.1007/s00285-013-0705-3. 4.1

Eldar Giladi, John Healy, Gene Myers, Chris Hart, Phillip Kapranov, Doron Lipson, Steven Roels, Edward Thayer, and Stan Letovsky. Error tolerant indexing and alignment of short reads with covering template families. *Journal of Computational Biology*, 17(10):1397–1411, 2010. doi: 10.1089/cmb.2010.0005. 1

Mathieu Giraud, Mikaël Salson, Marc Duez, Céline Villenet, Sabine Quief, Aurélie Caillault, Nathalie Grardel, Christophe Roumier, Claude Preudhomme, and Martin Figeac. Fast multiclonal clusterization of V(D)J recombinations from high-throughput sequencing. *BMC Genomics*, 15(409), 2014. doi: 10.1186/1471-2164-15-409. 4.2

Robert S. Harris. *Improved pairwise alignment of genomic DNA*. Ph.d. thesis, The Pennsylvania State University, December 2007. 1

Bernhard Haubold, Nora Pierstorff, Friedrich Möller, and Thomas Wiehe. Genome comparison without alignment using shortest unique substrings. *BMC Bioinformatics*, 6(123), 2005. doi: 10.1186/1471-2105-6-123. 4.2

Nils Homer, Barry Merriman, and Stanley F. Nelson. BFAST: An alignment tool for large scale genome resequencing. *PLoS One*, 4(11):e7767, 2009. doi: 10.1371/journal.pone.0007767. 1

John Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi and A. Paz, editors, *The Theory of Machines and Computation*, pages 189–196. Academic Press, New York, 1971. 3.2

Sebastian Horwege, Sebastian Lindner, Marcus Boden, Klas Hatje, Martin Kollmar, Chris-André Leimeister, and Burkhard Morgenstern. Spaced words and kmacs: Fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Research*, 42(W1):W7–W11, 2014. doi: 10.1093/nar/gku398. (document), 4, 4.2, 4.2.2, 5

Liang Huang. Dynamic programming algorithms in semiring and hypergraph frameworks. Technical report, University of Pennsylvania, Philadelphia, USA, November 2006. 3.3

Lucian Ilie, Silvana Ilie, and Anahita Mansouri Bigvand. SpEED: fast computation of sensitive spaced seeds. *Bioinformatics*, 27(17):2433–2434, 2011. doi: 10.1093/bioinformatics/btr368. 1

Lucian Ilie, Hamid Mohamadi, Geoffrey Brian Golding, and William F. Smyth. BOND: Basic OligoNucleotide Design. *BMC Bioinformatics*, 14(69), 2013. doi: 10.1186/1471-2105-14-69. 1

Thorsten Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer/Springer, 2002. doi: 10.1007/978-1-4615-0907-3. 4.1.1

Uri Keich, Ming Li, Bin Ma, and John Tromp. On spaced seeds for similarity search. *Discrete Applied Mathematics*, 138(3):253–263, 2004. doi: 10.1016/S0166-218X(03)00382-2. 1, 2

Szymon M. Kiełbasa, Raymond Wan, Kengo Sato, Paul Horton, and Martin C. Frith. Adaptive seeds tame genomic sequence comparison. *Genome Research*, 21(3):487–493, 2011. doi: 10.1101/gr.113985.110. 1, 5

Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of Bioinformatics and Computational Biology*, 3(3):527–550, 2005. doi: 10.1109/CSB.2004.135. 4.1

Gregory Kucherov, Laurent Noé, and Mikhail A. Roytberg. Multiseed lossless filtration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(1):51–61, 2005. doi: 10.1109/tcbb.2005.12. 1

Gregory Kucherov, Laurent Noé, and Mikhail A. Roytberg. A unifying framework for seed sensitivity and its application to subset seeds. *Journal of Bioinformatics and Computational Biology*, 4(2):553–569, 2006. doi: 10.1142/S0219720006001977. 1, 3.3, A.1, A.1

Gregory Kucherov, Laurent Noé, and Mikhail A. Roytberg. Subset seed automaton. In *Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA)*, volume 4783 of *LNCS*, pages 180–191, 2007. doi: 10.1007/978-3-540-76336-9_18. 3.2, 5

Gregory Kucherov, Laurent Noé, and Mikhail A. Roytberg. Iedera subset seed design tool. http://bioinfo.lifl.fr/yass/iedera.php, 2014. 3.2, 5

Chris-André Leimeister, Marcus Boden, Sebastian Horwege, Sebastian Lindner, and Burkhard Morgenstern. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, 30(14):1991–1999, 2014. doi: 10.1093/bioinformatics/btu177. (document), 4, 4.2, 4.2.2, 5

Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 564–575, 2002. 4.1

Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinfomatics*, 20(4):467–476, 2004. doi: 10.1093/bioinformatics/btg431. 4.1, 5

Ming Li, Bin Ma, Derek Kisman, and John Tromp. PatternHunter II: Highly sensitive and fast homology search. *Journal of Bioinformatics and Computational Biology*, 2(3):417–439, 2004. doi: 10.1142/S0219720004000661. 1, 4.2.2, 5

Hao Lin, Zefeng Zhang, Michael Q. Zhang, Bin Ma, and Ming Li. ZOOM! Zillions Of Oligos Mapped. *Bioinformatics*, 24(21):2431–2437, 2008. doi: 10.1093/bioinformatics/btn416. 1

Zongzhi Liu, Todd Z. DeSantis, Gary L. Andersen, and Rob Knight. Accurate taxonomy assignments from 16S rRNA sequences produced by highly parallel pyrosequencers. *Nucleic Acids Research*, 36(18):e120, 2008. doi: 10.1093/nar/gkn491. 4.2

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002. doi: 10.1162/153244302760200687. 4.1

Tobias Marschall, Inke Herms, Hans-Michael Kaltenbach, and Sven Rahmann. Probabilistic arithmetic automata and their applications. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(6): 1737–1750, 2012. doi: 10.1109/TCBB.2012.109. 1, 3.3, 5

Donald E. K. Martin. Coverage of spaced seeds as a measure of clumping. In *JSM Proceedings, Statistical Computing Section*, Alexandria, Virginia, 2013. American Statistical Association. (document), 3.1, 3.2, 3.3, 5

Donald E. K. Martin and Deidra A. Coleman. Distribution of clump statistics for a collection of words. *Journal of Applied Probability*, 48(4):901–1204, 2011. doi: 10.1239/jap/1324046018. 1, 5

Donald E. K. Martin and Laurent Noé. Faster exact probabilities for statistics of overlapping pattern occurrences. *Submitted to the Annals of the Institute of Statistical Mathematics (AISM)*, 2014. (document), 3.1, 3.2, 3.3, 5, A.1, A.2

Sebastian Maurer-Stroh, Vithiagaran Gunalan, Wing-Cheong Wong, and Frank Eisenhaber. A simple shortcut to unsupervised alignment-free phylogenetic genome groupings, even from unassembled sequencing reads. *Journal of Bioinformatics and Computational Biology*, 11(6):1343005, 2013. doi: 10.1142/S0219720013430051. 4.2

Charles D. Michener and Robert R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11(2): 130–162, June 1957. 4.2.1

Mehryar Mohri. *Handbook of Weighted Automata*, chapter Weighted Automata Algorithms, pages 213–254. Springer, 2009. doi: 10.1007/978-3-642-01492-5_6. 3.3

François Nicolas and Éric Rivals. Hardness of optimal spaced seed design. *Journal of Computer and System Sciences*, 74(5):831–849, 2008. doi: 10.1016/j.jcss.2007.10.001. 1

Grégory Nuel. Pattern Markov chains: optimal Markov chain embedding through deterministic finite automata. *Journal of Applied Probability*, 45:226–243, 2008. 3.3

Grégory Nuel. *Bioinformatics - Trends and Methodologies*, chapter Significance Score of Motifs in Biological Sequences. InTech, 2011. doi: 10.5772/18448. 1

Octave community. GNU Octave 3.8. http://www.gnu.org/software/octave/, 2014. 3.2

Taku Onodera and Tetsuo Shibuya. The gapped spectrum kernel for support vector machines. In *Proceedings of the International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, volume 7988 of *LNCS*, pages 1–15, 2013. doi: 10.1007/978-3-642-39712-7_1. (document), 4, 4.1, 5

Jean-Éric Pin. Tropical semirings. In J. Gunawardena, editor, *Idempotency*, volume 11 of *Publ. Newton Inst.*, pages 50–69, Bristol, 1998. Cambridge Univ. Press. 3.3

Ji Qi, Hong Luo, and Bailin Hao. CVTree: A phylogenetic tree reconstruction tool based on whole genomes. *Nucleic Acids Research*, 32(suppl 2):W45–W47, 2004. doi: 10.1093/nar/gkh362. 4.2

Kim R. Rasmussen, Jens Stoye, and Eugene W. Myers. Efficient q-gram filters for finding all ϵ-matches over a given length. *Journal of Computational Biology*, 13(2):296–308, 2006. doi: 10.1089/cmb.2006.13.296. 2

Mireille Régnier, Billy Fang, and Daria Iakovishina. Clump combinatorics, automata, and word asymptotics. In *Proceedings of the Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 2014. doi: 10.1137/1.9781611973204.6. 1, 5

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinfomatics*, 20(11):1682–1689, 2004. doi: 10.1093/bioinformatics/bth141. 4.1

Craige Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13:179–191, 1961. doi: 10.4153/CJM-1961-015-3. 4.2.3

Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In *Mathematical foundations of computer science*, volume 324 of *LNCS*, pages 107–120, 1988. doi: 10.1007/BFb0017135. 3.3

Gregory E. Simsa, Se-Ran Juna, Guohong A. Wua, and Sung-Hou Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106 (8):2677–2682, 2009. doi: 10.1073/pnas.0813249106. 4.2, 5

Valeri T. Stefanov, Stéphane Robin, and Sophie Schbath. Waiting times for clumps of patterns and for structured motifs in random sequences. *Discrete Applied Mathematics*, 155(6-7):868–880, 2007. doi: 10.1016/j.dam.2005.07.016. 1, 5

Pooj K. Stropea and Etsuko N. Moriyama. Simple alignment-free methods for protein classification: A case study from G-protein-coupled receptors. *Genomics*, 89(5):602–612, 2007. doi: 10.1016/j.ygeno.2007.01.008. 4.2

Susana Vinga. Editorial: Alignment-free methods in computational biology. *Briefings in Bioinformatics*, 15(3): 341–342, 2014. doi: 10.1093/bib/bbu005. 4.2

Susana Vinga and Jonas Almeida. Alignment-free sequence comparison - a review. *Bioinformatics*, 19(4):513–523, 2003. doi: 10.1093/bioinformatics/btg005. 4.2, 4.2.1

Jialiang Yang and Louxin Zhang. Run probabilities of seed-like patterns and identifying good transition seeds. *Journal of Computational Biology*, 15(10):1295–1313, 2008. doi: 10.1089/cmb.2007.0209. 5

Leming Zhou, Ingrid Mihai, and Liliana Florea. Spaced seeds for cross-species cDNA-to-genome sequence alignment. *Communications in Information and Systems*, 10(2):115–136, 2010. 1

A Seed coverage automaton size

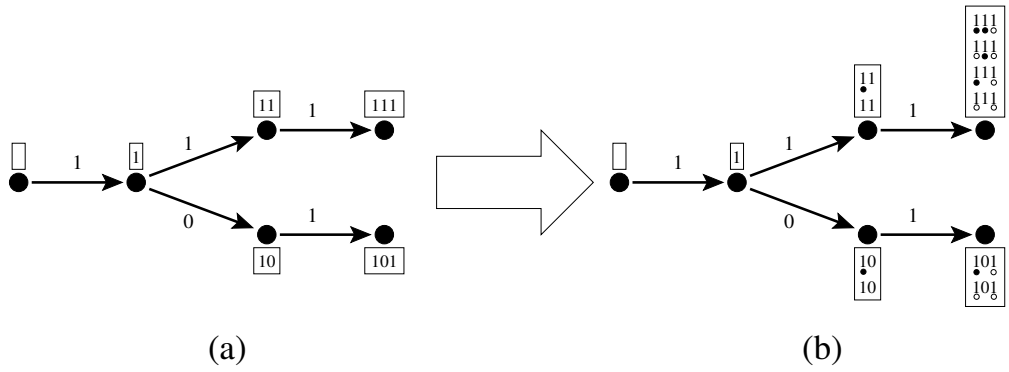
We consider in this part the size of the seed automaton. Given a seed of weight w and r jokers we are particularly interested in a bound for the size of the coverage automaton, as this can provide a limit on memory needed for future analyses.

In this section, we first solve the problem in the special case of a seed of the form $1*^r1$, before going to a more general case of a seed of weight w and r jokers, for which we show a more general (but less satisfying) upper bound.

A.1 Seed $1*^r1$ coverage automaton size

Figure 7: Moore multi-hit (a) and Moore coverage (b) automata size illustrated for the seed $\pi = 1 * 1$.

In boxes are set all the seed prefixes q that can be reached for the Moore multi-hit (a) and the Moore coverage (b) automata. Additionally, on the coverage automaton (b), for each prefix q, we have enumerated all the possible coverage strings c that are compatible to form $\langle {}^{q}_{c} \rangle$ states : this is done by substituting any non-covered 1 symbol of $\langle {}^{q}_{c} \rangle$ (but the last) by a possibly covered one $\overset{\bullet}{1}$ and, for final states, by considering newly covered positions $\overset{\circ}{1}$.



The $1*^r1$ seed family has already been shown to reach the multi-hit automaton size bound (Kucherov et al., 2006) : as a nightmare for the classical seed design tools, such seeds are good candidates to start with.

The *multi-hit* automaton size is, in the general case, of maximal size $(w + 1)2^r$ (Buhler et al., 2005, Kucherov et al., 2006). Moreover, for seeds of the form $1*^r1$, this size cannot be reduced further (Kucherov et al., 2006) : thus, $1*^r1$ always have multi-hit automata of size 3×2^r (illustrated in Figure 7 (a) where not all the transitions are shown).

The *coverage* automaton size for seeds of the form $1*^r1$ is respectively 4×3^r for the Moore automaton, and 3×3^r for the Mealy automaton.

Proof. We concentrate first on the Moore automaton. The set of states for the coverage *automaton can be easily deduced from the* multi-hit *automaton by considering, for each of the* multi-hit *prefixes q, all the possible coverages c that are compatible with the current prefix to form reachable $\langle {}^{q}_{c} \rangle$ states. Any prefix q may have any of its* 1*-positions (but the last) covered by a previous hit of a seed if this previous hit* **ends** *at this* 1*-position (illustrated by the dot symbols of Figure 7 (b) to mark* 1*-positions already covered). Moreover, it must be noticed that coverage of any* 1*-positions inside q can be chosen independently, by making/disabling a previous hit of a seed using its* **first** 1*-position (this position is not shown on the automaton, thus not overlapping the current prefix, and does not have any side effect). Thus all the possible* 1*-positions (but the last) of a given proper prefix q can be chosen independently with or without coverage. Thus, for any proper prefix q of length $l+1$ $(0 < l+1 < k)$ (q overlaps the first* must match *symbol, followed by $l = 1 \dots r$* joker *symbols of the seed ongoing hit)*

1. *the very first* 1 *symbol under a* must match *symbol can be covered or not (two possibilities :* 1 *or* $\overset{\bullet}{1}$*),*
2. *the next $l-1$ symbols under joker symbols can be independently chosen as* 0 *or* 1 *(three possibilities :* 0*,* 1 *or* $\overset{\bullet}{1}$*),*
3. *the very last symbol under the last joker symbol can be independently chosen as* 0 *or* 1 *(two possibilities). Note that this* 1*, as new, cannot be covered by a previous hit.*

For a given prefix q with $l = 1 \dots r$ jokers, there are thus $2 \times 3^{l-1} \times 2 = 4 \times 3^{l-1}$ possible $\langle {}^{q}_{c} \rangle$ states . Finally, the final states can be seen as prefixes q of length $k = r+2$, where the last 1 *is always newly covered (one choice :* $\overset{\circ}{1}$*), the r jokers can be any of* 0*,* 1 *or* $\overset{\bullet}{1}$ *(3 choices), and the very first* 1 *can be previously covered or newly covered (2 choices :* $\overset{\bullet}{1}$ *or* $\overset{\circ}{1}$ *when considering the Moore automaton), leading to $3^r \times 2$ final $\langle {}^{q}_{c} \rangle$ states. At the end, adding the initial state for $q = \epsilon$, and its next state (for $q =$ "1" corresponding to the first must match position of the seed which cannot be covered), gives :*

$$\underbrace{2}_{\text{initial state + next state}} + \underbrace{\sum_{l=1}^{r} 4 \times 3^{l-1}}_{\text{proper prefixes of length } 0 < l+1 < k} + \underbrace{2 \times 3^{r}}_{\text{last final states}}$$

$$=$$

$$4 \times 3^{r}$$

Such seeds $1*^{r}1$ *have thus a Moore coverage automata of size 4×3^r.*

Note that this size cannot be reduced. In other words, given any pair of states $\langle {q_a \atop c_a} \rangle$ *and* $\langle {q_b \atop c_b} \rangle$ *on this automaton, and starting (from each of these states) a walk by reading the same (given) string* u *:*

- *if* q_a *and* q_b *are different, then it is always possible to find one string* u *such that* only one of the two *walks reaches a final state (as done in Kucherov et al., 2006).*
- *otherwise, the coverages* c_a *and* c_b *must be different : it is then always possible to find one string* u *going to two final states that have a* different coverage *increment for the Moore automaton.*

We concentrate now on the Mealy automaton. The main difference with the Moore automaton is that suffixes of full length k *(that are final states of the Moore automaton) are not represented because coverage values are set on transitions, and not on states (see Martin and Noé, 2014).*

For the Mealy automaton of Martin and Noé (2014), and seeds of the form `1*`^{r}`1` *(of length* $k = r + 2$ *and weight* 2 *), there are* 2×3^l *proper prefixes* q *of length* $l + 1$ *(*$0 \leq l + 1 < k$*) :*

1. *the first symbol must be* `1`*, or* $\underset{\bullet}{1}$ *(two possibilities),*
2. *the next* l *symbols can be* `0`*,* `1` *or* $\underset{\bullet}{1}$ *(*3^l *possibilities).*

Adding the initial state, gives :

$$\underbrace{1}_{initial\ state} + \underbrace{\sum_{l=0}^{r} 2 \times 3^l}_{proper\ prefixes\ of\ length\ 0 \leq l+1 < k}$$

$$=$$

$$1 + 2 \times \frac{3^{r+1} - 1}{2}$$

$$=$$

$$3^{r+1}$$

This bound is reached for the same reasons of non-reducibility (applied on transition labels on Mealy, and not on final state labels as in Moore).

□

A.2 Coverage automaton size in the general case

Now consider a seed of span k with r jokers and of weight w ($w + r = k$). Following the previous section A.1, a similar reasoning gives a bound on the automaton size of $2^w \times 3^r$ for the Moore automaton and of $(2^w - 1) \times 3^r$ for the Mealy automaton.

Proof. We concentrate first on the Moore automaton. We respectively call r_j *and* w_j *the number of* joker *symbols and* must match *symbols for a given seed prefix of length* j ($r_j = \left|{}_j(\pi)\right|_*$, $w_j = \left|{}_j(\pi)\right|_1$, $r_j + w_j = j$). *We don't necessarily suppose that the seed starts and ends with a* must match *symbol. We will show that the number of states* $\langle {q \atop c} \rangle$ *such that* $|q|$ *and* $|c|$ *are* $\leq j$ *is at most* $2^{w_j} \times 3^{r_j}$, *by induction.*

- *This is first true for* $j = 0$, *because the* empty state *(also called the* initial state*)* $\langle {q \atop c} \rangle$ *where* $|q| = |c| = 0$ *is the only one that can match the empty seed prefix* ${}_0(\pi)$.

- *If we suppose that it is true for a given* i ($\#\{states \quad \langle {q \atop c} \rangle \quad \text{with} \quad |q| = |c| \leq i\} \leq 2^{w_i} \times 3^{r_i}$), *it can be now considered for* $j = i + 1$. *We split the demonstration for* j *in two parts :*

  1. *when* $|q| = |c| \leq i$, *by taking the set of the* $2^{w_i} \times 3^{r_i}$ *possible* $\langle {q \atop c} \rangle$ *states (induction hypothesis)*
  2. *otherwise, when* $|q| = |c| = j$, *by considering and adding to this set the states* $\langle {q \atop c} \rangle$ *that can be possibly reached. Two cases must then be considered :*
     (a) *if the last symbol* $\pi[j]$ *of the seed prefix* ${}_j(\pi)$ *is a* must match, *this symbol can only be compatible with a* `1` *on* $q[j]$ *(and this* `1` *cannot be covered by* $c[j]$, *as the last one being added).*
     (b) *if the last symbol* $\pi[j]$ *of the seed prefix* ${}_j(\pi)$ *is a* joker, *this symbol can be compatible either with a* `0` *or a* `1` *on* $q[j]$ *(which cannot be covered by* $c[j]$ *too) .*

     Considering now the prefix ${}_i(\pi)$ *preceding* $\pi[j]$, *we can see that :*

     - *the* w_i must match *symbols of* ${}_i(\pi)$ *are compatible with a* `1` *or a* $\underset{\bullet}{1}$ *(*2^{w_i} *possibilities),*
     - *the remaining* r_i jokers *of* ${}_i(\pi)$ *are compatible with a* `0`, *a* `1` *or a* $\underset{\bullet}{1}$ *(*3^{r_i} *possibilities).*

     Combining each of the cases (a) and (b) with the preceding prefix ${}_i(\pi)$ *gives* $1 \times 2^{w_i} 3^{r_i}$ *states for (a), or* $2 \times 2^{w_i} 3^{r_i}$ *states for (b), respectively, when* $|q| = |c| = j$.

  At the end, because (a) $w_j = w_i + 1$ *and* $r_j = r_i$, *or (b)* $w_j = w_i$ *and* $r_j = r_i + 1$ *otherwise, we can see that summing the number of states when* $|q| = |c| \leq i$ *and when* $|q| = |c| = j$ *gives the expected result* $2^{w_j} \times 3^{r_j}$, *for* non-final states.

It must be then noticed that, even for final states*, new symbols that have just been covered (*$\overset{\circ}{1}$*) are only replacing the non-covered ones (*1*) on a subset of the w fully determined positions given by the seed shape that are not yet covered (*$\overset{\bullet}{1}$*): they thus don't modify the recurrence when $j = |\pi|$ as they simply represent indicators to compute the coverage increment.*

We concentrate now on the Mealy automaton. Again, the main difference with the Moore automaton is that suffixes of full length k are not represented because coverage values are set on transitions, but one thing to consider is that the last symbol can be covered on the Mealy automaton (see Martin and Noé, 2014). By a similar reasoning, there are thus at most $\sum_{i=0}^{k-1=r+w-1} 3^{r_i} \times 2^{w_i}$ *states in the general case, and :*

$$\sum_{i=0}^{r+w-1} 3^{r_i} \times 2^{w_i} \leq \sum_{i=0}^{r} 3^i + 3^r \sum_{i=1}^{w-1} 2^i = 2^w 3^r - \frac{3^r + 1}{2} < 2^w 3^r$$

Note that if we suppose that the seed starts with a must match *symbol, then this bound can be reduced a little more :*

$$\sum_{i=0}^{r+w-1} 3^{r_i} \times 2^{w_i} \leq 1 + 2(\sum_{i=0}^{r} 3^i + 3^r \sum_{i=1}^{w-2} 2^i) = (2^w - 1)3^r$$

□

We notice in practice a much smaller size, and we suspect this bound more likely to be a $polynom(w, r) \times 3^r$ value, instead of exponential both in 2^w and 3^r. In the special case of symmetric seeds, we already have a very simple proof of this $polynom(w, r) \times 3^r$ bound. This is interesting, because experimentally, seeds of the form `11`^{u}`(*1`^{u}`)`^{r}`1` have been shown to give large coverage automata size.

Note even if not satisfying, the general result still improves on the only available "bound" proposed to date (in Benson and Mak, 2008) that can be estimated to be of order $w2^w4^r$.