

CHAPTER 1

COMPARING, RANKING AND FILTERING MOTIFS WITH CHARACTER CLASSES : APPLICATION TO BIOLOGICAL SEQUENCE ANALYSIS

MATTEO COMIN AND DAVIDE VERZOTTO

University of Padova, Italy

1.1 INTRODUCTION

In biology the notion of motif plays a central role for describing various phenomena. For example, protein functional motifs, like the ones contained in the PROSITE database [20], e.g. $[FY]DPC[LIM][ASG]C[ASG]$, are in general represented as motifs with character classes. These motifs are collected using semi-automatic procedures, nevertheless they are still manually verified.

The discovery of sequence motifs in proteins and genes is becoming increasingly important [2, 29]. Such motifs usually correspond to residues conserved during evolution due to some significant structural or functional role. Moreover the increasing availability of biological sequences, such as whole genomes, from next-generation sequencing technologies to new protein discoveries, has increased the need for automatic methods for their analysis and comparison.

In order to fill this gap, researchers have developed several approaches over the years. Typically these approaches follow in popular frameworks of motif discovery

and matching [4, 6, 17, 23, 26, 30, 41, 44]. Although such methods have been exhaustively consolidated and successfully tested on small genomes and set of proteins [12, 21, 25], the algorithm's outcome often exceeds the length of the original input [10, 36, 40], making it impossible to be handled and then interpreted for further analyses that require manual inspection. The main underlying difficulty of these frameworks is that, in order to model mutations and other evolutionary mechanisms, motifs have to account for some degree of variability. To this end motifs are often modeled to include undetermined symbols, '.' called *don't care*, e.g. *DPC.IM..C*, or classes of symbols, e.g. *[FY]DPC[LIM][ASG]C[ASG]*. Unfortunately the discovery and matching of such motifs can be prohibitive. In fact, even in a simple string there might be an exponential number of motifs with don't care symbols.

The rest of the chapter is organized as follows. In Section 1.2, we give a characterization of motifs with character classes, following with the notion of *motif priority* for comparing and ranking different motifs together. In Section 1.3, we introduce the concept of *underlying motifs* for filtering any set of motifs with character classes into a new set that is linear in size with respect to a reference sequence. We then present an algorithm to compute this new set exploiting the notions of Section 1.2. Finally, in Section 1.4 we discuss some preliminary results on the identification of signals in protein sequences by means of underlying motifs.

1.1.1 Ranking and Clustering Motifs

To cope with the combinatorial explosion of motifs, motif discovery tools must first shrink the search space. In this regard, a number of different techniques have been proposed over the past two decades. In general these techniques are based on the lexicographic composition of motifs [3, 5, 22, 28, 34] or on their statistical distribution [6, 33, 42]. Despite that, the number of motifs in output still remains in most cases intractable.

Moreover all motif discovery tools must ultimately rank the output, according to some measure of importance. For example, even if very sophisticated probabilistic scoring mechanisms are in place in a state-of-the-art tool such as Varun [4], there are cases where the direct interpretation of results is far from trivial. Let us consider, for instance, the worst case of [4] presented in Figure 1.1. The functionally relevant motif, shown in bold, appears at position 42, whereas almost all top motifs are somehow very similar. In order to filter out this output and enhance its readability, there are two main issues. On the one hand, most motifs are very similar in sequence and therefore they must be clustered together in some way. On the other hand, if we are interested in a specific region of the reference sequence, one would like to select the most important motif that appears in that region according to some rule. These two issues are in practice tightly related and they need to be addressed as one.

1.1.2 Ensemble Methods

In the last decade a number of ensemble methods have been developed [35, 43, 46]. The basic idea is that one can integrate the outcome of several motif discovery al-

Rank	z-score	Motif
1	2.84E+09	Y..L...C.[FYW]A.[STAH]R.P.FNE[STAH]K.I.F[STAH]M
2	8.28E+07	V-(1,3,4)G...S.[STAH]...N...L...Q-(4)[STAH]...L[DN]...[FYW].F...P...Q.A...I
3	5.55E+07	L-(2,3)F...Q...[STAH][STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
4	4.27E+07	L-(2,3)F...Q[STAH].[STAH][STAH]...S...[FYW].F.R.PD.Q.A...I
5	4.23E+07	L...L.[STAH].[STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
6	3.99E+07	LF-(3)Q...[STAH][STAH]...S[DN]...[FYW].F.R.PD.Q.A...I
7	3.38E+07	LF-(3)Q...[STAH][STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
8	3.38E+07	LF...Q...[STAH]-(4)L[DN]...[FYW].F.R.PD.Q[STAH].A...I
9	3.29E+07	I-(1)Q[STAH].[STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
10	3.29E+07	I.Q-(4)[STAH]...L[DN]...[FYW].F.R.PD.Q[STAH].A...I
11	3.29E+07	I.Q[STAH].[STAH]-(4)L[DN]...[FYW].F.R.PD.Q.A...I
12	3.10E+07	L...Q-(1,4)[STAH].[STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
13	2.77E+07	L[FYW]-(3)Q[STAH].[STAH]...L...[FYW].F.R.PD.Q.A...I
14	2.58E+07	L-(4)Q[STAH].[STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
15	2.30E+07	S[STAH]S-(2,4)L[DN]...[FYW].F.R.PD.Q[STAH].A...I
16	2.15E+07	L-(1,3,4)C...[FYW]A.[STAH]R.P.F.E.K.I.F.M
17	1.40E+07	F-(1)I.Q...[STAH][STAH]-(4)L[STAH]...[FYW].F.R.PD.Q.A...I
18	1.37E+07	L-(2,4)L...[STAH][STAH][STAH]-(3)LS...[FYW].F.R.PD.Q.A...I
19	1.02E+07	L..I-(1)Q...[STAH][STAH]...S...[FYW].F.R.PD.Q.A...I
20	8.65E+06	I-(1)Q...[STAH][STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
21	8.19E+06	S[STAH]-(1,2,3,4)L[DN]...[FYW].F.R.PD.Q[STAH].A...I
22	7.98E+06	Q-(3)[STAH][STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
23	6.82E+06	F-(3)Q...[STAH][STAH]...L[STAH]...[FYW].F.R.PD.Q.A...I
24	5.66E+06	A[STAH][STAH]-(2,3)L[DN]...[FYW].F.R.PD.Q.A...I
25	5.57E+06	F.I-(3)[STAH].[STAH]...L[STAH]...[FYW].F.R.PD.Q.A...I
26	5.18E+06	L.L-(4)Q...[STAH]...L-(1)DN]...[FYW].F.R.PD.Q.A...I
27	3.61E+06	L.L-(2)L...[STAH]...[STAH]...[STAH]...[FYW].F.R.PD.Q.A...I
28	3.48E+06	[STAH].[STAH]-(1,2,3)L[DN]...[FYW].F.R.PD.Q.A...I
29	3.17E+06	[STAH]...[STAH]...L[DN]...[FYW].F.R.PD.Q.A...I
30	2.47E+06	L...Q-(4)[STAH][STAH]...S...[FYW].F.R.PD.Q.A...I
31	2.43E+06	V-(1,3)N.L...I-(3)[STAH]...[STAH]...[STAH]...[FYW].F...PD.Q.A...I
32	2.22E+06	[STAH][STAH][STAH]-(1,2,3)LS...[FYW].F.R.PD.Q.A...I
33	2.06E+06	[STAH][STAH][STAH]...L...[FYW].F.R.PD.Q.A...I
34	2.03E+06	Y..L...C...A...R.P.F.E.K.I.F.I.F[STAH]
35	1.99E+06	I.Q...[STAH]-(1)[STAH]...L[DN]...[FYW].F...PD.Q.A...I
36	1.99E+06	I.Q-(1)[STAH]...[STAH]...L[DN]...[FYW].F...PD.Q.A...I
38	1.97E+06	F.I...[STAH]-(3)[STAH]...L[DN]...[FYW].F...PD.Q.A...I
40	1.97E+06	F.I-(3)[STAH].[STAH]...L[DN]...[FYW].F...PD.Q.A...I
41	1.91E+06	[STAH].[STAH].K-(1,4)P.FNE[STAH]K.I.F[STAH]M
42	1.72E+06	CC[FYW].C.C...[FYW]-(2,4)[DN]...[STAH]C..C
43	1.57E+06	[STAH]-(1,3,4)[FYW]A.[STAH]R.P.F.E.K.I.F.M
44	1.49E+06	A-(1,3)[STAH]...L[STAH]DN]...[FYW].F.R.PD.Q.A...I
45	1.36E+06	Q...[STAH][STAH]-(3)L[STAH]...[FYW].F.R.PD.Q.A...I
46	1.32E+06	I-(3)[STAH].[STAH][STAH]...S...[FYW].F.R.PD.Q.A...I
47	1.31E+06	[STAH][STAH]-(1,2,3,4)L[DN]...[FYW].F.R.PD.Q.A...I
48	1.24E+06	[STAH].[STAH][STAH]-(1,3)LS...[FYW].F.R.PD.Q.A...I
49	1.19E+06	[FYW]-(1,3,4)[STAH]...P.FNE[STAH]K.I.F[STAH]M
50	1.12E+06	L...[STAH]-(3)[STAH]...L[STAH]...[FYW].F.R.PD.Q.A...I

Figure 1.1 One of the functionally relevant motifs is shown in bold for G-protein coupled receptors family 3 (id PS00980). Output of Varun [4] for 25 sequences of about 25,000 amino acids each.

gorithms based on simple and efficient heuristics, to improve the ability of finding functional motifs in specific contexts. The ensemble methods actually rank all predicted motifs according to some scoring functions, and then report the top motifs. WebMotifs [35], ARCS-Motif [46], MotifMiner [11], and MotifVoter [43] assume that some of the motifs given by the consensus of several finders are likely to be functional motifs. They ultimately cluster all motifs and report only those from the best clusters. However, if none of the motifs from the individual finders can accurately capture the binding sites, the performance of the ensemble methods will suffer. Although these ensemble methods indeed help to improve the performance of motif finding, the improvement is usually not significant. For example, in Tompa's benchmark [39] and Escherichia coli datasets, the average sensitivity is only improved by 62 %, but the average precision is reduced by 15 %.

This relatively new topic deserves more attentions, along with its mathematical and combinatorial foundations. In this chapter we will discuss the basic problems behind motifs comparison and ranking. We will show how simple lexicographic properties

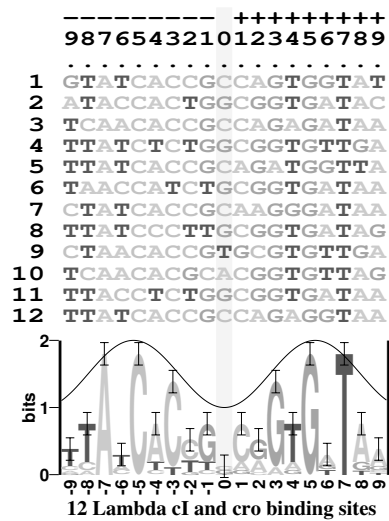


Figure 1.2 An example of sequence alignment logo taken from [37].

can help to select a small subset of motifs, therefore filtering the output of any finder to a more tractable size.

1.1.3 Motif Representation

In general a motif is represented by an ordered sequence of symbols and a set of locations where it appears. The motif, along with its list of occurrences, are usually extracted from one or more reference sequences over a fixed-length alphabet Σ .

To model mutations and other evolutionary events, a number of formalisms have been introduced. For example, position-specific scoring matrices (PSSMs, [7]) are widely used to model transcription factor binding sites, where each score represents the probability, or number of times, that a given symbol appears at a certain position. See Figure 1.2 for an example, including the multiple alignment of sequences that generates a PSSM.

Another popular model are character classes, like the IUPAC alphabet.¹ For protein sequences a number of clustering approaches have been proposed [14, 19, 20, 24]. Similarly, other protocols exist for DNA [13], and also for RNA. For example, the PROSITE motif for the Nickel-dependent hydrogenases large subunit signature is *RG[LIVMF]E.....[QESMP][RK].C[GR][LIVM]C*.

¹IUPAC stands for International Union of Pure and Applied Chemistry. The alphabet main reference is: Nomenclature committee of the International Union of Biochemistry (NC-IUB), "Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1984." Journal of Biological Chemistry 261, 13–17.

In this study we are interested in motifs with character classes. Most of the time, even if the classification, say of amino acids, might lead to a partition of symbols, in practical cases motifs show classes with non-empty intersections. For example, in the above motif there are intersections between classes, like that of $[RK]$ with $[GR]$, and also subsets, like $[LIVM]$ with respect to $[LIVMF]$. If the motifs in input contain only partitions of the original alphabet Σ , we can map the reference sequences into the new alphabet of symbols represented by the partitions, thus considerably reducing the number of candidate motifs to be analyzed [40]. In all other cases this will result in the combinatorial explosion of candidate motifs, that consequently affects the efficiency of matching and discovering new motifs. In this chapter we will set the basis to solve this fundamental problem.

1.1.4 Problem Statement

The main problem we want to address here is the following:

Given a reference sequence s , that might be the concatenation of multiple biological sequences, and a set of motifs with character classes \mathcal{M} , that is the outcome of one or more motif finders, we aim to reduce the set \mathcal{M} to a small number of ordered representative motifs, say \mathcal{U} , such that the size of \mathcal{U} is linear with respect to the reference sequence s .

This reduction will allow us to manage the new set easier, and will enhance its readability and interpretation. The general idea behind this work is to identify interesting lexicographic properties of motifs that are useful for the above problem. In particular, we will prove that some properties, along with the notion of *minimality* that will be defined in the next section, are capable to rank all possible motifs with character classes, and also to select a representative subset of motifs, called *underlying representative set* \mathcal{U} . Inherently, with the notion of underlying motifs we want to discard those motifs that rank lower than others if located in the same occurrences on the reference sequence s .

In the following we first characterize motifs with character classes defining the notion of minimality, that makes the motifs more specific with respect to the reference sequence s . Then, we define the notion of *motif priority*, that permits to compare and rank different motifs together. The priority is defined using properties like the length of a motif, its composition (accounting also for character classes), and the list of its occurrences. This motif priority rule has proved to be useful for filtering a set of biological motifs \mathcal{M} . In particular the mentioned lexicographic properties, combined with the list of occurrences of a motif, have been specifically designed for proteins by analyzing the results of [4, 12]. Nevertheless, we will prove that we are able to rank all minimal motifs using our motif priority rule, and we will show that it is possible to filter out a set of motifs \mathcal{M} into a new set of linear size with respect to s .

In summary, using our simple motif priority rule we can prove that any set \mathcal{M} is *sub-ordered*. Moreover, using the notion of minimality we can map any set of motifs \mathcal{M} into its *minimal representative set* $\mu(\mathcal{M})$, that is *totally ordered*. This lays the foundations for filtering any set of motifs into a set that is linear in size with respect to the reference sequence s . As a proof of concept we will present an algorithm that has

time complexity $O(n|\mu(\mathcal{M})| \log |\mu(\mathcal{M})|)$, or $O(n \max\{|\mathcal{M}|, |\mu(\mathcal{M})| \log |\mu(\mathcal{M})|\})$ whenever the computation of minimal motifs is required, where n is the size of the sequence s .

We believe that such an approach can drastically reduce the number of candidate motifs to be analyzed in genomics and proteomics, and can enhance their readability and interpretation. In this section we present a series of results on protein families that support the validity of the theoretical findings. Preliminary experiments on protein families show very good performance as a filter to reduce the number of motifs while keeping the most important ones.

1.2 MOTIFS WITH CHARACTER CLASSES: A CHARACTERIZATION

We define a *string* as a sequence of zero or more symbols from an alphabet Σ . The set of all strings over Σ is denoted by Σ^* . The length of a string x is defined as the number of its symbols, denoted by $|x|$, and the i -th symbol of x is denoted by x_i , where $1 \leq i \leq |x|$.

For example, in genomics Σ corresponds to the set of nucleotides $\{A, C, G, T\}$ or $\{A, C, G, U\}$, respectively, for DNA and RNA, while in proteomics Σ corresponds to the set of the 20 amino acids: $\{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$. Then, a symbol σ from Σ is called *solid character*.

Throughout the chapter a string s composed of solid characters from Σ will be used as reference sequence. Conversely, motifs will be represented by a more expressive alphabet, as explained in the following definitions.

Definition 1.1 (*Character class*) Let R_Σ be a set of equivalence relations over Σ . A character class is a subset of Σ consisting of two or more equivalent solid characters with respect to some relation in R_Σ .

For instance, the hydrophobicity property can partitioning all 20 amino acids into disjoint classes. In this regard a number of different hydrophobicity scales have been developed over the years [8]. A hydrophobicity scale thus generates an equivalence relation $R \in R_\Sigma$ between solid characters, such that each class of the resulting quotient set Σ/R can be defined as a character class. It is possible to define several character classes over Σ , according to the equivalence relations in R_Σ . In particular, we consider the set of maximal character classes on R_Σ , i.e., those classes that are not a proper subset of other classes. This argument will be further discussed in Subsection 1.2.1.

Given τ maximal character classes $\{C_i\}_1^\tau$ on Σ , $\bigcup_{i=1}^\tau 2^{C_i}$ represents all non-empty subsets of the classes.

Definition 1.2 (*Pattern*) A pattern with character classes, or simply a pattern, is a string defined on $\Sigma \cup (\bigcup_{i=1}^\tau 2^{C_i})$.

In the literature, a pattern with character classes is also called *degenerate pattern* or *indeterminate string* [1, 16, 27, 31, 32, 38, 45]. Given a pattern $p = p_1 p_2 \dots p_k$

j	1	2	3
p	a	$[a, c, d]$	$[b, e]$

j	1	2	3	4	5	6	7	8	9	10
s	a	a	b	e	a	d	b	a	c	e
p										

Figure 1.3 Example of a pattern p with character classes, along with its occurrences 1, 5, 8 in the string s .

of length k , each symbol p_j , with $1 \leq j \leq k$, is either a solid character or a subset of a character class C . In the latter case, we write p_j by means of square brackets, as the notation of PROSITE [20]: $p_j = [\sigma_1, \sigma_2, \dots, \sigma_{r_j}]$.

Definition 1.3 (*Pattern occurrence*) A pattern $p = p_1 p_2 \dots p_k$ of length k is said to occur at a location l of a string s , with $1 \leq l \leq n$, if $s_{l+j-1} \in p_j$ for each $1 \leq j \leq k$.

Figure 1.3 shows an example of pattern and pattern occurrences, where $s = aabeadbace$, $\Sigma = \{a, b, c, d, e\}$, and the character classes are $C_1 = \{a, c, d\}$ and $C_2 = \{a, b, e\}$. In this case, $p = a[a, c, d][b, e]$ is a pattern of length $k = 3$ that occurs at locations 1, 5, 8 of s .

Hereafter we will assume that we are given a string s on Σ of length n , a set of character classes $\{C_i\}_1^r$, with $r = \max_{1 \leq i \leq \tau} \{|C_i|\}$, and a positive integer q , $2 \leq q \leq |s|$, called *quorum*. A *motif* m is then defined as a pair (p, \mathcal{L}_m) , where p is a pattern of length k that occurs at the locations given by \mathcal{L}_m , and \mathcal{L}_m is a set of at least q locations of s . For instance, as of Figure 1.3, $m = (p, \mathcal{L}_m)$ is a motif with pattern $p = a[a, c, d][b, e]$ of length $k = 3$ and location list $\mathcal{L}_m = \{1, 5, 8\}$ of cardinality $\nu = 3$. More formally:

Definition 1.4 (*Motif, Location list*) We say that $m = (p, \mathcal{L}_m)$ is a motif with pattern $p = p_1 p_2 \dots p_k$ and location list $\mathcal{L}_m = (l_1, l_2, \dots, l_\nu)$ if and only if all of the following hold:

- (i) the length of m , denoted by $|m|$, is $|p| = k \geq 2$ symbols,
- (ii) for each location $l \in \mathcal{L}_m$, p occurs at l in s , and $|\mathcal{L}_m| = \nu \geq q$, and
- (iii) there does not exist a location $l' \notin \mathcal{L}_m$ such that p occurs at l' in s (\mathcal{L}_m is complete).

We suppose now that an oracle has given us a set \mathcal{M} of interesting motifs lying on the string s . Intuitively this set may be unusually large, with the number of elements that may reach up to $O(\sum_{k=2}^{n-q+1} (\min\{\tau 2^r + |\Sigma|, 2^{|\Sigma|}\})^k) = O((\min\{\tau 2^r, 2^{|\Sigma|}\})^n) = O(2^{|\Sigma|n})$ motifs. Thus, we want to characterize the whole set of motifs \mathcal{M} with *representative motifs* that have to be $O(n)$ in number and dimensions, in order to filter

out the redundant information shared by some motifs, and enhance their readability and interpretation.

Next sections lay the foundations for finding these particular representative motifs, that we will call *underlying*, which, in loose terms, are the most important for some regions of the string s , according to the rules we will define in the following.

1.2.1 On Transitive Properties of Character Classes

Previously, we have introduced the concept of character class consisting of two or more solid characters, obtained through some equivalence relation on Σ . In this case every character class is seen as an independent element wherewith building motifs. Then, since for patterns there is no distinction between a character class and its subsets, we consider maximal character classes as input for our problems.

Typical is the case where the considered relations of equivalence between solid characters, R_Σ , are not mutually transitive, since they might partition Σ in different ways [9, 41, 44]. For example, if σ_1, σ_2 are two solid characters belonging to a character class C_1 , $\sigma_1, \sigma_3 \in C_2$, and $\sigma_2, \sigma_3 \in C_3$, then it could be the case that $\sigma_1, \sigma_2, \sigma_3$ do not belong to a common class C_4 .

Consider, for simplicity, that R_Σ is mutually transitive on Σ . Then, the set of character classes partitions Σ , and we can map each solid character on Σ into either its maximal character class or itself. Call $R_\Sigma(\cdot)$ this function. Accordingly, we can map $s = s_1 s_2 \dots s_n$ into $R_\Sigma(s)$, given by $R_\Sigma(s) = R_\Sigma(s_1) R_\Sigma(s_2) \dots R_\Sigma(s_n)$. In this way every pattern becomes a pattern on the new restricted alphabet given by $R_\Sigma(\Sigma)$, that is $\Sigma \cup \{C_i\}_1^\tau$. Moreover, the discovery of this type of patterns, called in the literature *exact patterns*, has been consolidated over the years [15, 18, 23, 25, 40]. In short, if R_Σ is mutually transitive we can find a set of classes that is a partition of Σ . Then we can map the reference sequence on this new alphabet and use standard pattern techniques. As a consequence, every motif can be uniquely identified by its length and location list, and we can conclude that, for this particular case, there is no need of the notion of *minimality* that will be introduced in the next subsection.

More attention, however, deserves the general case where the transitivity does not hold. As stressed above, the main issue is that the character classes might not be a partition of Σ . Because of this, the motifs are not uniquely identified by their length and location list. In other words, there may exist two motifs m and m' lying on s that have equal length and location list, but different patterns p and p' , because of different character classes that compose the corresponding symbols. Since classical motif discovery algorithms find motifs over the restricted alphabet $\Sigma \cup \{C_i\}_1^\tau$, because of its practicality and ease of computation, this may lead to a large number of motifs with same location list. Thus, in the next subsection we study the notion of *minimality* specifically designed to address this issue.

j	1	2	3
p	$[a, c, d]$	$[a, c, d]$	$[a, b, e]$
$\mu(p)$	a	$[a, c, d]$	$[b, e]$

Figure 1.4 Example of minimal representation $\mu(p)$ of a pattern p , with reference string $s = aabeadbace$.

1.2.2 Minimal Motifs and Motif Priority

Definition 1.5 (*Minimal representation*) Given a motif m of length k , the minimal representation of its pattern p is a pattern $\mu(p)$ of length k with symbols $\mu(p)_j = \bigcup_{l \in \mathcal{L}_m} s_{l+j-1}$, for $1 \leq j \leq k$.

Fact 1.1 *The minimal representation of a pattern is unique.*

Remark 1.1 *Since $\mu(p)$ is more specific than p , that is, $\mu(p)$ cannot have more occurrences than p in s , then the list of occurrences of $\mu(p)$ must be the same as for p .*

Let $\mu(m) = (\mu(p), \mathcal{L}_m)$ be the minimal representation of a motif m with pattern p , then Remark 1.1 suggests us that $\mu(m)$ is consistent with the definition of motif, i.e., the location list $\mathcal{L}_{\mu(m)}$ is complete:

Definition 1.6 (*Minimal motif*) *The minimal representation of m , given by $\mu(m) = (\mu(p), \mathcal{L}_m)$, is called minimal motif.*

Computing the minimal representation of a motif is useful when a motif is defined on $\Sigma \cup \{C_i\}_1^r$, rather than on specific characters. To have a more concrete idea about this concept, Figure 1.4 shows an example of minimal representation $\mu(m)$ of a motif $m = ([a, c, d][a, c, d][a, b, e], \{1, 5, 8\})$, with pattern $\mu(p)$ and p respectively, where the reference string is $s = aabeadbace$ and the character classes are $C_1 = \{a, c, d\}$ and $C_2 = \{a, b, e\}$. In this case we say that $m' = \mu(m) = (a[a, c, d][b, e], \{1, 5, 8\})$ is a minimal motif.

Let \mathcal{M} be a set of motifs with character classes lying on the string s . From Fact 1.1, each motif $m \in \mathcal{M}$ has a unique minimal representation $\mu(m)$. Thus, one can easily check that the equivalence given by $\mu(m) = \mu(m')$ is an equivalence relation. The representatives of the induced partitions are the respective unique minimal motifs.

Fact 1.2 *Let us map all the motifs in \mathcal{M} into the set of their minimal representations $\mu(\mathcal{M})$, where each motif $m \in \mu(\mathcal{M})$ is the minimal representation $m = \mu(m')$ of some motif $m' \in \mathcal{M}$. Then, the set of motifs \mathcal{M} is partitioned into equivalence classes by the binary relation of equality between minimal representations.*

We call $\mu(\mathcal{M})$ the *minimal representative set* of \mathcal{M} . Since mapping \mathcal{M} into $\mu(\mathcal{M})$ could mean a drastic reduction in the number of motifs, this is in practice a first step for filtering the original motifs in \mathcal{M} .

Remark 1.2 *Two motifs in \mathcal{M} or, equivalently, in $\mu(\mathcal{M})$ may have the same location list. However, two minimal motifs of equal length must have different location lists.*

As a consequence of Remark 1.2, every minimal motifs in $\mu(\mathcal{M})$ can be uniquely identified by its length and location list within the minimal representative set. We can now define an important property of motifs with character classes, the *composition*, that is the sum of the number of characters within the symbols of a motif.

Definition 1.7 (*Composition of a pattern, Composition of a motif*) *The composition of a pattern p of length k is defined as $c(p) = \sum_{j=1}^k |p_j|$. The composition of a motif $m = (p, \mathcal{L}_m)$, denoted by $c(m)$, is defined as the composition of its pattern $c(p)$.*

For instance, given two patterns $p = a[a, c, d][b, e]$ and $p' = [a, d]b[a, b]$, their composition is $c(p) = 6$ and $c(p') = 5$, and we say that the latter is somehow more specific than p . Moreover, the composition of the motif $m = (a[a, c, d][b, e], \{1, 5, 8\})$ is equal to $c(p)$, that is 6.

A fundamental point is the following definition of *priority* between motifs, a means for comparing different motifs. Here we give priority to motif length, then to motif composition, and finally to motifs that appear first in s .

Definition 1.8 (*Motif priority*) *A motif m of length k has priority over a motif m' of length k' , denoted $m \rightarrow m'$, if and only if either $m = m'$ or*

- (1) $k > k'$, or
- (2) $k = k'$ and $c(m) < c(m')$, or
- (3) $k = k'$, $c(m) = c(m')$, and $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\}$ when both minima exist.

For instance, consider $s = aabeadbace$, $m = (a[a, c, d][b, e], \{1, 5, 8\})$, and $m' = ([a, d]b[a, b, e], \{2, 6\})$. Then, m has priority over m' , written $m \rightarrow m'$, because of equal length and composition, but with different order of appearance in s : $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} = 1 < \min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\} = 2$. If two distinct motifs m and m' have equal length and composition, but either $\mathcal{L}_m \subseteq \mathcal{L}_{m'}$ (that includes the equality case) or $\mathcal{L}_{m'} \subset \mathcal{L}_m$, then we consider m and m' *incomparable*, otherwise they are comparable. Nevertheless, we will solve this issue along with Theorem 1.2.

Given our binary relation, called *motifpriority*, we want to prove that some intuitive properties hold:

Definition 1.9 (*Acyclicity*) *A binary relation R over a set \mathcal{M} is defined to be acyclic if there does not exist a chain of distinct elements of \mathcal{M} , $m_1, m_2, m_3, \dots, m_t$, such that $m_1 R m_2, m_2 R m_3, \dots, m_{t-1} R m_t$, and also $m_t R m_1$ (cycle).*

Definition 1.10 (*Sub-ordered set, Totally ordered set*) *Let R be a binary relation over a set \mathcal{M} . Then, \mathcal{M} is said to be sub-ordered with respect to R , if for all distinct elements m_1, m_2, \dots, m_t in \mathcal{M} :*

- (i) $m_1 R m_1$ (reflexivity), and
- (ii) $\neg(m_1 R m_2 \text{ and } m_2 R m_1)$ (antisymmetry), and
- (iii) there does not exist a cycle between m_1, m_2, \dots, m_t (acyclicity).

If also (iv) $(m_1 R m_2 \text{ or } m_2 R m_1) \forall m_1, m_2 \in \mathcal{M}$ (totality), then m_1 and m_2 are said to be comparable, and \mathcal{M} is totally sub-ordered.

Furthermore, \mathcal{M} is totally ordered if R is reflexive, antisymmetric, total, and transitive, that is, (v) $(m_1 R m_2 \text{ and } m_2 R m_3) \Rightarrow m_1 R m_3$.

One can prove that condition (i) and (iii) of Definition 1.10 are sufficient to define a sub-ordered set, nevertheless we have listed all the three properties of sub-ordered sets for clarity and completeness.

Lemma 1.1 *A set \mathcal{M} is totally ordered under a binary relation R if and only if it is totally sub-ordered.*

Proof: It is straightforward that transitivity and antisymmetry, together, imply acyclicity, i.e., if \mathcal{M} is totally ordered, then \mathcal{M} is also totally sub-ordered. Consider a chain of distinct elements in \mathcal{M} : $m_1 R m_2, m_2 R m_3, \dots, m_{t-1} R m_t$. It follows that $m_1 R m_t$ for the transitivity, and that $m_t R m_1$ cannot hold since R is antisymmetric.

We can also easily check that acyclicity and totality, together, imply transitivity. Consider the same chain of elements as above. Since for the acyclicity $m_t R m_1$ cannot hold, then, for the totality, $m_1 R m_t$. Therefore the transitivity holds, and the definitions of totally sub-ordered and totally ordered coincide. ■

Now, before showing that some of these properties hold, we need to prove two preliminary results for the motif priority rule. At first, we observe that:

Fact 1.3 *The binary relation of motif priority is reflexive and antisymmetric, since properties (1), (2), and (3) of Definition 1.8 are strictly defined.*

Lemma 1.2 *(Paired occurrences) Let m and m' be two distinct motifs with $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, such that both minima exist, and define j to be $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$. Then, the occurrences of m and m' at positions less than j must be the same. Furthermore, we call these occurrences paired.*

Proof: We have to prove that, in case (3) of Definition 1.8, the occurrences of m and m' less than j are identical.

If m has an occurrence less than j that is not in $\mathcal{L}_{m'}$, then $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\} < j$, that is impossible by hypothesis. Conversely, if m' has occurrences less than j that are not in \mathcal{L}_m , then $\min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\} < j = \min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ contradicts again our assumptions. Thus, the occurrences of the two motifs less than j must coincide, and we call them paired.

Finally, by assumptions, it trivially holds that $j \notin \mathcal{L}_{m'}$. ■

Lemma 1.3 *Let $m_1 \rightarrow m_2, m_2 \rightarrow m_3, \dots, m_{t-1} \rightarrow m_t$ be a chain of distinct motifs of equal length and composition. Then, either $m_1 \rightarrow m_t$ or $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$ holds.*

Proof: We will prove the statement by induction on t . Let the basis be $t = 2$. In this case, the chain $m_1 \rightarrow m_2$ coincides with the result. We show now that, for $t > 2$, if it holds either $m_1 \rightarrow m_{t-1}$ or $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$, then either $m_1 \rightarrow m_t$ or $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$ holds.

Assume $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$. Define j to be $\min\{l \mid l \in \mathcal{L}_{m_{t-1}} \setminus \mathcal{L}_{m_t}\}$, and observe that it exists since $m_{t-2} \rightarrow m_{t-1}$, by property (3) of Definition 1.8. It follows that $j \in \mathcal{L}_{m_1}$ and, from Lemma 1.2, that the occurrences of m_{t-1} and m_t that are less than j are paired. Hence, these occurrences are also shared with m_1 . Since $j \notin \mathcal{L}_{m_t}$, then either $m_1 \rightarrow m_t$, because j makes the difference between \mathcal{L}_{m_1} and \mathcal{L}_{m_t} , or m_1 and m_t are incomparable. In the latter case, $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_1}$ and $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_t}$, together, imply that $\mathcal{L}_{m_{t-1}} \subset \mathcal{L}_{m_t}$; that is impossible since m_{t-1} and m_t are comparable by hypothesis. This means that the latter case must be $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$.

Conversely, assume $m_1 \rightarrow m_{t-1}$. Define j as above, and $j' = \min\{l \mid l \in \mathcal{L}_{m_1} \setminus \mathcal{L}_{m_{t-1}}\}$. It holds that $j \notin \mathcal{L}_{m_t}$, as already observed, and that $j' \neq j$ because of $j' \notin \mathcal{L}_{m_{t-1}}$. Then, by Lemma 1.2, the occurrences of m_{t-1} and m_t that are less than j , are paired, and the same it is for occurrences of m_1 and m_{t-1} less than j' . In this way, if m_1 and m_t are comparable, then there exists an occurrence in \mathcal{L}_{m_1} that appears first in s with respect to \mathcal{L}_{m_t} : if $j' < j$, the occurrences of m_1, m_{t-1} , and m_t less than j' are paired together, and thus j' makes the difference; otherwise $j' > j$, and the occurrences of m_1, m_{t-1} , and m_t less than j are paired together, and hence $j \in \mathcal{L}_1$ makes the difference. Alternatively, if m_1 and m_t are incomparable, then it must be the case of $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$. Otherwise, $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_t}$ would imply that the occurrences of m_1 equal to or less than j' are also shared with m_t , which leads to $m_t \rightarrow m_{t-1}$; that is impossible. ■

Theorem 1.1 *Any set of motifs \mathcal{M} is sub-ordered with respect to the binary relation of motif priority.*

Proof: We have to prove that the relation of motif priority is reflexive, antisymmetric, and acyclic. The first two properties are stated in Fact 1.3. Now, following the work of Lemma 1.3, we can prove that the acyclicity holds too. First, observe that length and composition are intrinsic properties of the single motif, and thus monotone functions. If all motifs $m, m' \in \mathcal{M}$ have different length or composition, then, by definition of motif priority, it is always true that either $m \rightarrow m'$ or $m' \rightarrow m$ holds, and a cycle can never exist because of different lengths and/or compositions.

Alternatively, consider a chain of distinct motifs $m_1 \rightarrow m_2, m_2 \rightarrow m_3, \dots, m_{t-1} \rightarrow m_t$ of equal length and composition. In this case we must use property (3) of Definition 1.8 to compare the motifs together. From Lemma 1.3, it follows that a cycle of motif priority between any chain of distinct motifs is again impossible, and therefore the acyclicity holds. Furthermore, since there may exist a triad of motifs m_1, m_2, m_3 of equal length and composition, such that $m_1 \rightarrow m_2$ and $m_2 \rightarrow m_3$, but $\mathcal{L}_{m_3} \subset \mathcal{L}_{m_1}$, then the motif priority rule is definitely not transitive on \mathcal{M} . ■

Note that the non-decision on some binary comparisons finally discards the relations of totality and, as seen above, of transitivity. For instance, consider $s = aabeadbace$, $C_1 = \{a, c, d\}$, $C_2 = \{a, b, e\}$, and the motifs $m = ([a, d][a, b][a, e], \{2, 6\})$ and $m' = ([a, d][b, e][a, e], \{2, 6\})$ with the same list of occurrences: in short, $|m| = |m'| = 3$, $c(m) = c(m') = 6$, and $\mathcal{L}_m = \mathcal{L}_{m'}$. This means that we are not able to compare m and m' using the motif priority rule. Another example is given by $m_1 = (a[a, c, d][b, e], \{1, 5, 8\})$, $m_2 = ([b, e]a[a, c, d], \{4, 7\})$, and $m_3 = ([a, b][c, d][b, e], \{5, 8\})$. In this case, $m_1 \rightarrow m_2$ and $m_2 \rightarrow m_3$, but $m_1 \rightarrow m_3$ does not hold, since $\mathcal{L}_{m_3} \subset \mathcal{L}_{m_1}$. The issue is that m and m' may not be minimal motifs. In the following we set the basis to solve this problem.

Theorem 1.2 *Given any set of motifs \mathcal{M} , its minimal representative set $\mu(\mathcal{M})$ is totally ordered under the binary relation of motif priority.*

Proof: According with Theorem 1.1, any set of motifs \mathcal{M} is sub-ordered under the motif priority rule; thus, also the set of minimal motifs $\mu(\mathcal{M})$ is sub-ordered. Following the work of Lemma 1.1, we have to prove that the totality holds on this new set $\mu(\mathcal{M})$, i.e., every pair of minimal motifs must be comparable under motif priority. In other words, if $m, m' \in \mu(\mathcal{M})$, either $m \rightarrow m'$ or $m' \rightarrow m$ must hold. To this end the proof of Lemma 1.1 tells us that length and composition are intrinsic properties of motifs, and thus every motif is comparable with another one in case of different length or composition.

From Remark 1.2 we have that $\mathcal{L}_m \neq \mathcal{L}_{m'}$ for two minimal motifs m and m' of equal length, and therefore we have, without loss of generality, two scenarios to consider: $\mathcal{L}_m \not\subseteq \mathcal{L}_{m'}$ and $\mathcal{L}_{m'} \not\subseteq \mathcal{L}_m$, and $\mathcal{L}_m \subset \mathcal{L}_{m'}$. In the former case, if we consider $\min\{l \mid l \in \mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ and $\min\{l \mid l \in \mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, then both minima exist and are different from each other. Hence either $m \rightarrow m'$ or $m' \rightarrow m$ holds if it is the case of equal length and composition of m and m' , and the minimum between the two sets is either in \mathcal{L}_m or in $\mathcal{L}_{m'}$, respectively. Conversely, in the latter case, since m and m' are minimal motifs and the respective location lists are complete, the respective patterns of m and m' must be different. Therefore $\mathcal{L}_m \subset \mathcal{L}_{m'} \Rightarrow c(m) < c(m')$, and hence $m \rightarrow m'$. Accordingly, for any set of minimal motifs under motif priority, the totality holds. From Lemma 1.1 we can conclude that any set of minimal motifs is totally ordered under the motif priority rule. ■

As a consequence of Theorem 1.2, all minimal motifs can be compared and ranked. We can further observe that, for property (2) of motif priority, every minimal motif has priority over all other motifs in \mathcal{M} within its equivalence class, which is given by the paradigm of minimal representation.

Now it is clear that any set of motifs \mathcal{M} can be mapped into its minimal representative set, $\mu(\mathcal{M})$, and that we can build a measure of total order over this set, in particular using the motif priority rule, that otherwise would not be possible on the original set \mathcal{M} .

1.3 FILTERING BY MEANS OF UNDERLYING MOTIFS

In order to exploit the output of a motif discovery algorithm, our prior knowledge \mathcal{M} , and enhance its readability and interpretation, we aim to identify particular representative motifs of \mathcal{M} that have to be $O(n)$ in number and dimensions, where n is the length of the reference string s .

Let R be a binary relation of priority between motifs. The objective is to select the most priority motifs in \mathcal{M} for each location of s , according to R . If a motif m is selected, we discard all motifs with less priority that lie on the occurrences of m . The problem can be seen as a regional problem, in which we have to select some motifs for each considered region of the string s . These regions could be transcription factor binding sites or coding sequences if we consider genomes, otherwise functional regions if we consider a set of protein sequences. More formally:

Definition 1.11 (*Region*) A region $E_{j,x}$ of s is a set of x consecutive locations $\{j, j+1, \dots, j+x-1\}$ corresponding to the symbols $s_j s_{j+1} \dots s_{j+x-1}$, namely a substring of s , where $1 \leq j$ and $(j+x-1) \leq |s|$.

Definition 1.12 (*Living in a region*) We say that a motif m of length k lives in the region $E_{j,x}$ if there exists a location $l \in \mathcal{L}_m$ such that $(E_{l,k} \cap E_{j,x}) \neq \emptyset$.

Furthermore, we say that every motif m completely lives in the regions defined by its occurrences.

Definition 1.13 (*Tied occurrence*) Let m be a motif of length k . Then, we say that an occurrence l of m is tied to a motif m' , if m' lives in $E_{l,k}$ and $m' R m$. Otherwise, we say that l is untied from m' .

Definition 1.14 (*Underlying representative set, Underlying motif*) Let \mathcal{M} be a set of motifs that lie on the string s , and let u be a positive integer called underlying quorum. A set of motifs $\mathcal{U} \subseteq \mathcal{M}$ is said to be an underlying representative set of s if and only if:

- (i) every motif m in \mathcal{U} , called underlying motif, has at least u untied occurrences from any other motif in \mathcal{U} , and
- (ii) there does not exist a motif $m \in \mathcal{M} \setminus \mathcal{U}$ such that m has at least u untied occurrences from all motifs in \mathcal{U} .

In other words, given a string s and the information about all considered motifs \mathcal{M} , an underlying motif is a particular representative of some regions of s . Furthermore, considering the underlying quorum u as a fixed integer, it follows that Definition 1.14 of underlying representative set converges to one and only one set of motifs \mathcal{U} , under certain conditions:

Theorem 1.3 Let \mathcal{M} be a sub-ordered set of motifs with respect to a binary relation R . Then, there exists an underlying representative set $\mathcal{U} \subseteq \mathcal{M}$, and it is unique.

Proof: We show first that a set \mathcal{U} , absolving the two conditions of Definition 1.14, exists. If \mathcal{M} is sub-ordered, then there does not exist a cycle of priority between some distinct motifs m_1, m_2, \dots, m_t in \mathcal{M} (acyclicity). Consider, without loss of generality, $m_1 R m_t, m_2 R m_t, \dots, m_{t-1} R m_t$, such that no other motif has priority over m_t . This means that, for each region $E_{l,k}$ of s where m_t completely lives, either m_t or some other motifs in $\{m_1, m_2, \dots, m_{t-1}\}$ can have an untied occurrence in $E_{l,k}$; thus they must belong to some set of underlying motifs, say \mathcal{U} . Furthermore, any other motif m_{t+1} living in $E_{l,k}$, but with no priority relation with respect to m_t , can also be an underlying motif in \mathcal{U} . Similarly, the occurrences of $\{m_1, m_2, \dots, m_{t-1}\}$ living in $E_{l,k}$ must respect that rule. Since no cycle of priority is admitted, then, for all regions of s given by the occurrences of some motif m in \mathcal{M} , either m is in \mathcal{U} or there are some other motifs in \mathcal{U} that have untied occurrences from m and that cover those regions. In conclusion, there must be a set of underlying motifs \mathcal{U} .

In the opposite direction, we can prove that \mathcal{U} is unique. Let us assume that there exists two distinct underlying representative sets \mathcal{U}_1 and \mathcal{U}_2 . Then, for both sets, there exists at least one motif that is not in the other set, otherwise condition (ii) of Definition 1.14 does not hold. Suppose $m_1 \in \mathcal{U}_1 \setminus \mathcal{U}_2$. Since $m_1 \notin \mathcal{U}_2$, then, again for condition (ii), there must be an occurrence of m_1 in \mathcal{U}_1 , say the region E_{l_1, k_1} , that in \mathcal{U}_2 is covered by some other motif m_2 with higher priority than m_1 , such that $m_2 \notin \mathcal{U}_1$. For similar reasons, if m_2 is not in \mathcal{U}_1 , then there must be another motif $m_3 \in \mathcal{U}_1$ with higher priority than m_2 . Since \mathcal{M} is sub-ordered, then $m_3 \neq m_1$. Finally, as the number of motifs in \mathcal{U}_1 and \mathcal{U}_2 is finite, then the two sets must coincide. ■

The following result is a consequence of the proof of Theorem 1.3:

Proposition 1.1 *If \mathcal{M} is totally ordered, then each underlying motif m has priority over all other underlying motifs in \mathcal{U} in at least u regions of s .*

From this proposition, we have that, if R is a total relation over \mathcal{M} , then the number of underlying motifs on a string s is $O(n)$.

Theorem 1.4 *Let \mathcal{M} be a totally ordered set under the binary relation R . Then, the number of underlying motifs in \mathcal{U} is $\leq \lfloor n/2 \rfloor$, independently of the size of \mathcal{M} .*

Proof: From Proposition 1.1, every untied occurrence does not overlap with the untied occurrences of some other motifs in \mathcal{U} , and thus for every motif m in \mathcal{U} its untied occurrences will not overlap with those of other motifs. Since every motif m has length $k \geq 2$, then m must cover at least $2 + u - 1$ characters of s , obtaining at most $n/(2 + u - 1) \leq n/2$ possible different motifs in \mathcal{U} , independently of the size of \mathcal{M} . Furthermore, $|\mathcal{U}| \leq \lfloor n/(2u) \rfloor$ in case of non-overlapping occurrences for each of the motifs in \mathcal{U} . ■

Let us now consider the *dimensions* of a set of motifs as the number of characters necessary to specify all its elements. This number coincides with the sum of compositions of all motifs. For instance, $m = (p, \mathcal{L}_m)$, with $p = a[a, c, d][b, e]$, is a motif with 6 characters and composition $c(m) = 6$. Let r be the maximum number of characters allowed in a motif symbol, i.e., $r = \max_{1 \leq i \leq \tau} \{ |C_i| \}$ as introduced in Section 1.2. Then:

Corollary 1.1 *Given a string s of length n and a totally ordered set \mathcal{M} , any underlying representative set \mathcal{U} over \mathcal{M} has a total number of symbols, or total length, $\leq \lfloor n/2 \rfloor$ and dimensions no greater than $|\Sigma|n$.*

Proof: Following the proof of Theorem 1.4, all underlying motifs in \mathcal{U} can be placed on their untied occurrences in s without overlaps with other motifs. Therefore, the total number of symbols for the whole set of motifs \mathcal{U} is $\sum_{m \in \mathcal{U}} |m| \leq \lfloor n/2 \rfloor$. Hence \mathcal{U} has dimensions $\leq r \lfloor n/2 \rfloor \leq |\Sigma|n$. ■

For example, consider a scheme that encodes every solid character on Σ plus the two square brackets ‘[’ and ‘]’, in a simple injective way. Then, the dimension of each encoding would be at most $\lceil \log(|\Sigma| + 2) \rceil$. Therefore we can represent the symbols of a motif using at most $r \lceil \log(|\Sigma| + 2) \rceil$ bits per symbol. If we encode every underlying motif with this scheme, then the overall space needed to store all motifs in \mathcal{U} would be at most $r \lceil \log(|\Sigma| + 2) \rceil \lfloor n/(2 + u - 1) \rfloor$ bits, or $r \lceil \log(|\Sigma| + 2) \rceil \lfloor n/(2u) \rfloor$ bits in case of non-overlapping occurrences for each of the motifs in \mathcal{U} . This corresponds, in loose terms, to the result presented in Corollary 1.1. In conclusion, considering r as a constant, since it is bound by $|\Sigma|$, any underlying representative set \mathcal{U} has linear dimensions with respect to the size of the string s .

To summarize the results of the previous sections, we have that every set of motifs \mathcal{M} , that could be the output of some motif discovery algorithm, can be transformed into its minimal representative set $\mu(\mathcal{M})$. This is the basis to enable the comparison of all motifs in $\mu(\mathcal{M})$ by means of the notion of motif priority, that captures the lexicographical power of a motif. One can prove the validity of a series of properties that are fundamental for the global comparison of motifs. On this setting we can define the subset of underlying motifs, that are, for some locations of the reference string s , the most important. In the next subsection we discuss an algorithm to find the set of underlying motifs, while in Section 1.4 we will describe a series of experiments to prove the validity of the latter.

1.3.1 An Algorithm for Filtering a Set of Motifs into its Underlying Representative Set

Here we describe an application of motif priority and underlying motifs. We present an algorithm that filters a set of motifs \mathcal{M} lying on a string s into its underlying representative set \mathcal{U} .

Let R be a binary relation of priority. In the following we show how to build an underlying representative set \mathcal{U} in case of a totally ordered set \mathcal{M} under R . The purpose is to select the most representative motifs with character classes in \mathcal{M} , according to R and to the definition of underlying motif. In practice, we filter out those motifs in \mathcal{M} that rank lower than others if located in the same regions of s . From Corollary 1.1, since \mathcal{M} is totally ordered, it follows that \mathcal{U} has linear dimensions.

Consider now the following algorithm for filtering the motifs in \mathcal{M} into the underlying representative set $\mathcal{U} \subseteq \mathcal{M}$:

Algorithm 1.1

1. Rank all motifs in \mathcal{M} using the binary relation R .
2. Initialize \mathcal{U} to an empty set, and iteratively select a motif m from \mathcal{M} following the rank given by the priority.
3. At each step, if m has at least u untied occurrences from all motifs already in \mathcal{U} :
4. - add m to \mathcal{U} and store the regions of s in which m completely lives,
5. - otherwise, discard m .

The above algorithm is presented as a proof of concept. It is unclear if a more efficient implementation exists, or whether it is optimal. In this context we just analyze the correctness and the computational complexity of this algorithm. At first, we observe that the overall correctness follows from the proof of Theorem 1.3. Let n be the size of the string s , and let r be the maximum allowed size of a character class. If R corresponds to the motif priority rule, as of Subsection 1.2.2, in step 1 we may first preprocess the motifs in input, computing their lengths and compositions, in $O(rn|\mathcal{M}|)$ time. Then we sort all the motifs in \mathcal{M} , that are totally ordered, in descending order. Considering the worst case, when we compare two motifs, we must also compare their lists of occurrences. Since for some motif m , $|\mathcal{L}_m|$ is $O(n)$, then the comparison of two ordered lists of occurrences costs $O(n)$ (see the pseudocode presented in the function COMPAREMOTIF). Thus, overall the first step costs $O(n|\mathcal{M}|\log|\mathcal{M}| + rn|\mathcal{M}|)$.

The main cycle, line 2, selects the top motifs from \mathcal{M} and checks if they could be inserted in the set of underlying motifs \mathcal{U} . This loop can be stopped when no other motif can be added to the set \mathcal{U} .

After adding the first motif in the rank to \mathcal{U} , we check for untied occurrences of the next motif m' , having length k' . For each motif m that has been selected by our algorithm in \mathcal{U} , we store the occurrences of m in a vector of booleans $\Gamma[1, n]$, that represents all the locations of the string s . The value $\Gamma[i]$ is set to TRUE if the location i is tied to some motif m in \mathcal{U} . This means that, if m is in \mathcal{U} , then $\forall l \in \mathcal{L}_m$ we store the value TRUE at the locations of Γ that correspond to $E_{l,k}$, i.e. $\Gamma[l, l+k-1]$. Using the vector Γ , in the third instruction we have to check, for all $l' \in \mathcal{L}_{m'}$, whether some of the locations of $\Gamma[l', l'+k'-1]$, corresponding to the region $E_{l',k'}$, are set to TRUE. If it is the case, we say that l' is tied to some other motif with higher priority with respect to m' ; otherwise, l' is untied. Accordingly, if m' has at least u untied occurrences, we add m' to the set \mathcal{U} ; otherwise we discard m' . This step is detailed in the function CHECKFORUNTIEDOCCURRENCES. The checking for untied occurrences of a motif m , using the additional vector Γ , thus costs $O(n)$.

In the fourth instruction we have to update the vector Γ with all locations of the newly added underlying motif m . This can be done by scanning all the occurrences of m , and, $\forall l \in \mathcal{L}_m$, updating the values $\Gamma[l, l+k-1]$ with no backtracking. To this end, we use a variable δ that takes the maximum between $l+k-1$ and the next occurrence l' of m . This procedure is described in the function STORECOVERAGE.

Again this step takes $O(n)$ in time. In conclusion, the whole cycle of instructions 2–5 has a total cost of $O(n|\mathcal{M}|)$.

Since a motif m added to \mathcal{U} has higher priority than the motifs considered at the iterations that follow m , as of the second instruction, then the following invariant holds: at each stage, condition (i) of Definition 1.14 is satisfied. Hence, from Theorem 1.3, \mathcal{U} is the unique underlying representative set over \mathcal{M} .

In summary, the total complexity of Algorithm 1.1 is dominated by the first term $O(n|\mathcal{M}| \log |\mathcal{M}| + rn|\mathcal{M}|)$. Moreover, if we consider r to be a constant as above, then the total complexity of the algorithm becomes $O(n|\mathcal{M}| \log |\mathcal{M}|)$. For completeness, below you can find the pseudocode of the described implementation of Algorithm 1.1, that we called MOTIFFILTERING.

Finally, in most cases we need first to compute the minimal representative set $\mu(\mathcal{M})$, before ranking the motifs by means of the priority rule. This further process costs $O(n^2|\mathcal{M}| + rn|\mathcal{M}|)$. Nevertheless, since we consider r as a constant, and typically the location lists of motifs have non-overlapping occurrences, that is a realistic assumption for most applications in genomics and proteomics, then in practice the complexity of computing $\mu(\mathcal{M})$ becomes $O(n|\mathcal{M}|)$. In this regard, filtering the minimal motifs in $\mu(\mathcal{M})$ ultimately takes $O(n|\mu(\mathcal{M})| \log |\mu(\mathcal{M})|)$ time.

MOTIFFILTERING

Input: a sequence s , a list of motifs \mathcal{M} , an underlying quorum u ,
a binary relation R

Output: the underlying representative set \mathcal{U}

```

1  $n \leftarrow \text{length}(s)$ 
2  $\Gamma \leftarrow$  a vector of  $n$  booleans set to FALSE, representing  $E_{1,n}$ 
3  $\text{sort}(\mathcal{M}, R)$ 
4 for each motif  $m = (p, \mathcal{L}_m)$  in  $\mathcal{M}$ 
5     do  $\text{test} \leftarrow \text{CHECKFORUNTIEDOCCURRENCES}(m, u, \Gamma)$ 
6     if  $\text{test}$  is TRUE
7         then  $\Gamma \leftarrow \text{STORECOVERAGE}(m, \Gamma)$ 
8          $\mathcal{U} \leftarrow \mathcal{U} \cup \{m\}$ 
9 return  $\mathcal{U}$ 

```

STORECOVERAGE

Input: a motif m , a vector of booleans Γ

Output: the new vector of booleans Γ

```

1  $k \leftarrow \text{length}(m)$ ,  $\text{delta} \leftarrow 0$ 
2 for each location  $l$  in  $\mathcal{L}_m$ 
3     do  $\text{delta} \leftarrow \max\{\text{delta}, l\}$ 
4     store TRUE in the locations  $E_{\text{delta}, l+k-1}$  of  $\Gamma$ 
5      $\text{delta} \leftarrow l + k$ 
6 return  $\Gamma$ 

```

CHECKFORUNTIEDOCCURRENCES

Input: a motif m , an underlying quorum u , a vector of booleans Γ
 Output: the value TRUE, if the number of untied occurrences
 of m is at least u ; the value FALSE, otherwise

```

1  $k \leftarrow \text{length}(m)$ ,  $\text{delta} \leftarrow 0$ 
2 for each occurrence  $l$  in  $\mathcal{L}_m$ 
3   do  $\text{count} \leftarrow 0$ ,  $\text{untied} \leftarrow \text{TRUE}$ 
4      $\text{delta} \leftarrow \max\{\text{delta}, l\}$ 
5     for each location  $i$  in  $E_{\text{delta}, l+k-1}$  such that
6        $(l+k-1) \leq |\Gamma|$ 
7       do if  $\Gamma_i$  is TRUE
8         then  $\text{untied} \leftarrow \text{FALSE}$ 
9         break for
10    if  $\text{untied}$  is TRUE
11      then  $\text{count} ++$ ,  $i ++$ 
12      if  $\text{count} \geq u$ 
13        then return TRUE
14 return FALSE

```

COMPAREMOTIFS

Input: two minimal motifs m, m'
 Output: BEFORE, if $m \rightarrow m'$; AFTER, if $m' \rightarrow m$;
 EQUAL, otherwise

```

1 if  $m = m'$ 
2   then return EQUAL
3 if  $\text{length}(m) > \text{length}(m')$ 
4   then return BEFORE
5 if  $\text{length}(m) < \text{length}(m')$ 
6   then return AFTER
7 if  $c(m) < c(m')$ 
8   then return BEFORE
9 if  $c(m) > c(m')$ 
10  then return AFTER
11  $\triangleright$  At this stage, using the paired property described in
12   Lemma 1.2, we can simply compare the two ordered
13   lists of occurrences without any further check:
14  $i \leftarrow 0$ 
15 while TRUE
16   do if  $\mathcal{L}_m.\text{get}(i) < \mathcal{L}_{m'}.\text{get}(i)$ 
17     then return BEFORE
18   if  $\mathcal{L}_m.\text{get}(i) > \mathcal{L}_{m'}.\text{get}(i)$ 
19     then return AFTER
20    $i ++$ 
21 return EQUAL

```

1.4 EXPERIMENTAL RESULTS AND DISCUSSION

In this section we discuss the ability of underlying motifs to efficiently capture meaningful information. A general problem in genome and proteome research is the identification of signals represented by means of motifs. Several motif discovery tools proved to be useful to analyze biological sequences in different contexts. Here we first collect the motifs in output from one of these tools, say a set of motifs \mathcal{M} . Then we present some preliminary results, in order to support the theoretical properties shown in the above sections of the chapter.

More generally there are two types of scenarios where the notion of underlying motifs might be useful. The first case is when a region of interest has already been identified, so that it is possible to analyze and select only those motifs that are underlying with respect to that particular region, without considering the whole set of motifs. Another possible application is the case where we just want to filter all motifs in \mathcal{M} , looking at the whole sequence.

In this context we present some results for the latter scenario. We take as input \mathcal{M} the set of motifs extracted using the tool Varun [4]. The benchmark consists of two protein families for which Varun successfully extracts the functional motifs, as reported in the PROSITE database [20]. In particular, we consider the following protein families:

1. Nickel-dependent hydrogenases (id PS00508). These are enzymes that catalyze the reversible activation of hydrogen, and are further involved in the binding of nickel. The family is composed by 22 sequences of about 23,000 amino acids. One of the most functionally significant motifs is detected by Varun in the top three out of 4,150 extensible motifs.
2. Coagulation factors 5/8 type C domain (FA58C) (id PS01286). This family is composed by 40 sequences of about 46,432 amino acids. Varun places one of the most important structural and functional motifs in this family in the top two motifs out of 80,290 extensible motifs.

To prove that the information captured by the underlying motifs is relevant with respect to the protein family under examination, we devised two kinds of tests. In the first test we compare the original set of motifs in input, \mathcal{M} , with the set of underlying motifs in output \mathcal{U} , using global measures. In the second set of experiments we test the ability to filter meaningful motifs and retain the functional ones.

In the first set of experiments we use Varun to extract motifs from the above families of protein sequences, for different quorums q . In all these experiments the quorum refers to the number of occurrences in the concatenation of all sequences of a certain family. Then, we use the extracted motifs as input \mathcal{M} to our algorithm, presented in Subsection 1.3.1, in order to compute the underlying motifs \mathcal{U} . Three observations must be made at this point:

- the motifs extracted by Varun, $m \in \mathcal{M}$, are with character classes and no gaps;
- before using Algorithm 1.1, we compute the set of minimal motifs $\mu(\mathcal{M})$;

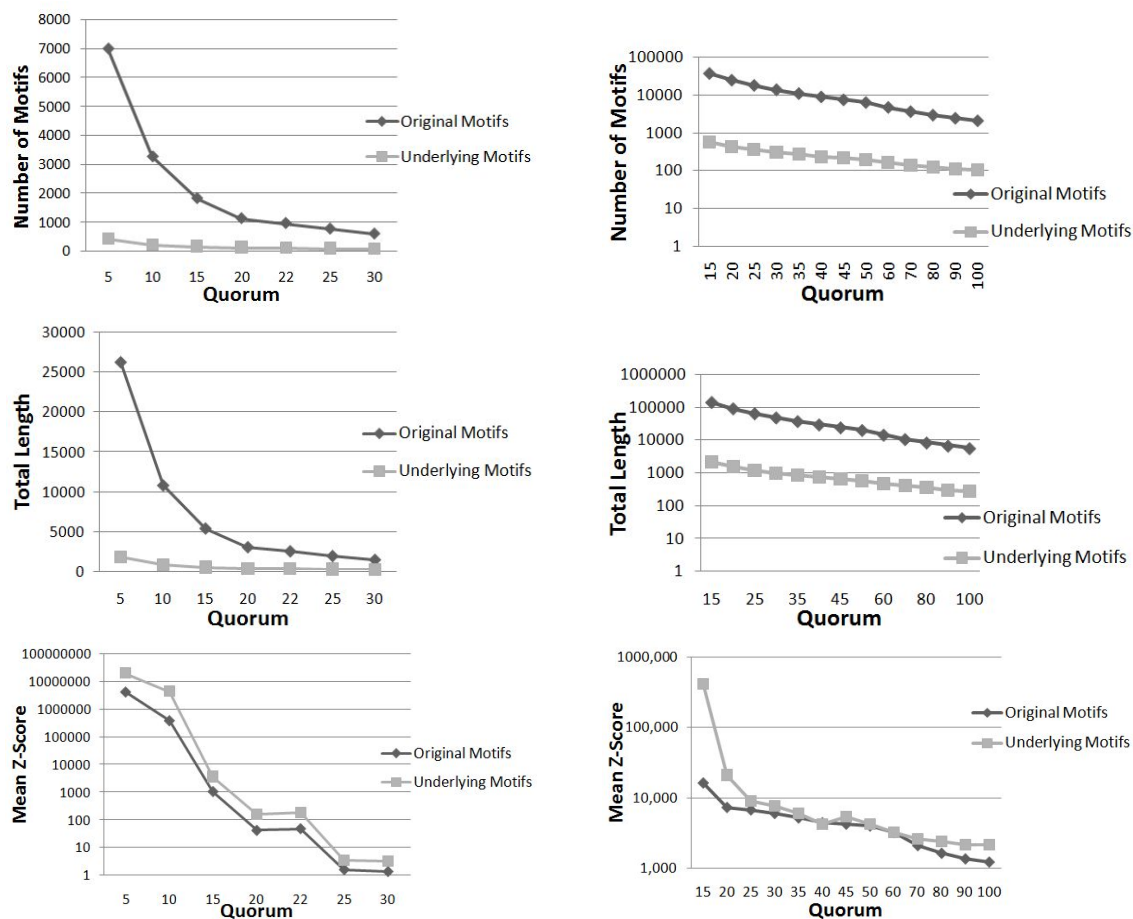
- in our experiments we set the quorum for underlying motifs u , to be the same as the original motifs in input, thus $u = q$.

For both sets of motifs \mathcal{M} and \mathcal{U} we compute some statistics. Figure 1.5 shows the number of motifs, the sum of lengths of all motifs and their mean Z-Score. The Z-Score is computed employing the same formula reported in [4]. As expected, the number of underlying motifs is always much smaller than the number of original motifs. A similar conclusion can be drawn for the sum of lengths. More important, for small quorums the total length of original motifs exceeds the length of sequences in both families, indicating that the original motifs are even larger than the set of sequences under examination. Moreover, as seen above, the sum of lengths of underlying motifs is always bounded by the length of sequences. These first two measures indicate that, not only the number, but also the total length of underlying motifs, is much smaller than the original motifs. Thus the filtering process is space-efficient.

Another important measure is the mean Z-Score of \mathcal{M} and \mathcal{U} . The mean Z-Score is a global measure that captures the average quality of motifs in a set. In Figure 1.5 we can see that, for all quorums, the average Z-Score of underlying motifs is always greater than those of original motifs, and in most cases the difference is one or two order of magnitude. To summarize, this first test confirms that number and span of underlying motifs is much more manageable than the original set, and also that their average quality is improved.

Once we have verified that the notion of underlying is a suitable filter, in a second series of experiments we test the ability to retain meaningful motifs. To this end, we employed the motifs obtained from the previous experiments to test the presence of functional motifs, as reported in the PROSITE database. In particular, the first family, Nickel-dependent hydrogenases (id PS00508), contains two functional motifs : $Ni_1 = RG[FILMV]E.....[EMPQS][KR].C[GR][ILMV]C$ and $Ni_2 = [FY]D[IP][CU][AII MV][AGS]C$. Similarly, the second family, coagulation factors 5/8 type C domain (FA58C) (id PS01286), shares the functional motifs: $Fa_1 = [FWY][ILV].[AFILV][DEGNST].....[FILV].[IV].[ILTV][KMQT]G$ and $Fa_2 = [LM]R.[EG][ILPV].GC$. For each set of motifs, \mathcal{M} and \mathcal{U} , we compute the maximum similarity between each motif in the set and the two functional motifs. The similarity between two motifs m and m' is the number of shared characters, including character classes, in the best alignment of m versus m' , without considering indels. Tables 1.1 and 1.2 summarize the maximum similarity, for different quorums, of \mathcal{M} and \mathcal{U} with each functional motif of the two protein families. The second and third columns report the maximum similarity for the set of underlying motifs divided by the similarity of the original set. For example, in the first row of Table 1.1 the quorum is 5. In this case, the maximum similarity of \mathcal{M} with the functional motif Ni_1 is 26. The same value is obtained also for the corresponding set of underlying motifs \mathcal{U} , thus indicating that the motif Ni_1 is retained with the same degree of accuracy.

The values presented in Table 1.1 and 1.2 confirm that, in most cases, the functional motifs, that are present in \mathcal{M} , are also selected in the set of underlying motifs with a similar accuracy. Ultimately, although a more comprehensive experimental setting



(a) Nickel-dependent hydrogenases (id PS00508) (b) Coagulation factors 5/8 type C domain (id PS01286)

Figure 1.5 Total number, sum of lengths, and mean Z-Score of the motifs extracted using Varun and their corresponding underlying motifs, for two protein families.

Quorum	Max Sim Ni_1 (underlying/original)	Max Sim Ni_2 (underlying/original)
5	26/26	9/12
10	18/18	12/12
15	11/11	9/12
20	9/9	12/12
22	9/9	12/12
25	6/6	6/6
30	6/6	6/6

Table 1.1 Performance of underlying motifs on the family Nickel-dependent hydrogenases.

Quorum	Max Sim Fa_1 (underlying/original)	Max Sim Fa_2 (underlying/original)
15	11/12	11/12
20	11/12	12/12
25	12/12	8/10
30	10/12	8/10
35	10/12	9/10
40	10/12	8/8
45	12/12	8/8
50	12/12	8/8
60	10/10	8/8
70	10/10	8/8
80	9/10	8/8
90	9/10	8/8
100	9/10	8/8

Table 1.2 Performance of underlying motifs on the family Coagulation factors 5/8 type C domain.

is desirable, this set of preliminary experiments support the validity of the theoretical results presented in the previous sections and also prove its effectiveness.

1.5 CONCLUSION

In this chapter we have studied motifs with character classes, introducing basic properties for the comparison and ranking of motifs. We have proved several theoretical results that support the validity of these fundamental properties. Most important, our motif priority rule along with the notion of underlying motifs proved to be valuable for the analysis of biological sequences. Preliminary experiments on protein families have shown very good performance as a filter to reduce the number of motifs in output, while keeping the most important ones.

Acknowledgments. Matteo Comin was partially supported by the Ateneo Project of the University of Padova and by the CaRiPaRo Project. Davide Verzotto was partially supported by “Fondazione Ing. Aldo Gini,” while visiting the University of California, Riverside.

REFERENCES

1. K. Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16:1039–1051, 1987.
2. A. Apostolico. Pattern discovery and the algorithmics of surprise. In *Proceedings of the NATO Advanced Study Institute on Artificial Intelligence and Heuristic Methods for Bioinformatics*, pages 111–127, 2002.
3. A. Apostolico, M. Comin, and L. Parida. Conservative extraction of over-represented extensible motifs. *Bioinformatics*, 21(Suppl.1):i9–i18, 2005. Proceedings of the Thirteenth International Conference on Intelligent Systems for Molecular Biology (ISMB 2005).
4. A. Apostolico, M. Comin, and L. Parida. VARUN: Discovering extensible motifs under saturation constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):752–762, October–December 2010.
5. A. Apostolico and L. Parida. Incremental paradigms of motif discovery. *Journal of Computational Biology*, 11:15–25, 2004.
6. T. L. Bailey, N. Williams, C. Mischel, and W. W. Li. MEME: Discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Research*, 34(Web-Server-Issue):369–373, 2006.
7. I. E. Ben-Gal, A. Shani, A. Gohr, J. Grau, S. Arviv, A. Shmilovici, S. Posch, and I. Grosse. Identification of transcription factor binding sites with variable-order Bayesian networks. *Bioinformatics*, 21(11):2657–2666, 2005.
8. K. M. Biswas, D. R. DeVido, and J. G. Dorsey. Evaluation of methods for measuring amino acid hydrophobicities and interactions. *Journal of Chromatography A*, 1000(1-2):637–655, 2003. A Century of Chromatography 1903–2003.
9. A. Chakravarty, J. M. Carlson, R. S. Khetani, C. E. DeZiel, and R. H. Gross. SPACER: Identification of cis-regulatory elements with non-contiguous critical residues. *Bioinformatics*, 23(8):1029–1031, 2007.
10. A. Chattaraj and L. Parida. An inexact-suffix-tree-based algorithm for detecting extensible patterns. *Theoretical Computer Science*, 335(1):3–14, 2005.
11. M. Coatney and S. Parthasarathy. MotifMiner: A general toolkit for efficiently identifying common substructures in molecules. In *Proceedings of the Third IEEE Symposium on Bioinformatics and BioEngineering (BIBE 2003)*, Washington, DC, USA, 2003. IEEE Computer Society.
12. M. Comin and D. Verzotto. Classification of protein sequences by means of irredundant patterns. *BMC Bioinformatics*, 11(Suppl.1):S16, 2010. Selected articles from the Eighth Asia-Pacific Bioinformatics Conference (APBC 2010).
13. A. Cornish-Bowden. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic acids research*, 13(9):3021–3030, May 1985.

14. C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, 2001.
15. M. Federico and N. Pisanti. Suffix tree characterization of maximal motifs in biological sequences. *Theoretical Computer Science*, 410(43):4391–4401, 2009.
16. K. Fredriksson and S. Grabowski. Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. *Information Retrieval*, 11:335–357, 2008.
17. M. C. Frith, N. F. W. Saunders, B. Kobe, and T. L. Bailey. Discovering sequence motifs with arbitrary insertions and deletions. *PLoS Computational Biology*, 4(5):e1000071, 2008.
18. D. Gusfield. *Algorithms on strings, trees, and sequences: Computer science and computational biology*. Cambridge University Press, New York, NY, 1997.
19. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89(22):10915–10919, 1992.
20. N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, B. Cuče, E. de Castro, C. Lachaize, P. Langendijk-Genevaux, and C. Sigrist. The 20 years of PROSITE. *Nucleic Acids Research (Database issue)*, 36:D245–D249, 2008.
21. K. L. Jensen, M. P. Styczynski, I. Rigoutsos, and G. N. Stephanopoulos. A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1):21–28, 2006.
22. H. Jiang, Y. Zhao, W. Chen, and W. Zheng. Searching maximal degenerate motifs guided by a compact suffix tree. In N. Back, I. R. Cohen, A. Lajtha, J. D. Lambris, R. Paoletti, and H. R. Arabnia, editors, *Advances in Computational Biology*, volume 680 of *Advances in Experimental Medicine and Biology*, pages 19–26. Springer New York, 2010.
23. R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC 1972)*, pages 125–136, 1972.
24. I. Ladunga and R. F. Smith. Amino acid substitutions preserve protein folding by conserving steric and hydrophobicity properties. *Protein Engineering*, 10(3):187–196, 1997.
25. C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.
26. N. D. Mendes, A. C. Casimiro, P. M. Santos, I. Sá-Correia, A. L. Oliveira, and A. T. Freitas. MUSA: a parameter free algorithm for the identification of biologically significant motifs. *Bioinformatics*, 22(24):2996–3002, 2006.
27. G. Navarro and M. Raffinot. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. *Journal of Computational Biology*, 10(6):903–923, 2003.
28. L. Parida. *Algorithmic Techniques in Computational Genomics*. Courant institute of mathematical sciences, New York University, 1998.
29. L. Parida. *Pattern Discovery in Bioinformatics: Theory and Algorithms*. Chapman and Hall/CRC Mathematical and Computational Biology, 2007.
30. N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot. RISOTTO: fast extraction of motifs with mismatches. In J. R. Correa, A. Hevia, and M. A. Kiwi, editors, *Proceeding of Seventh Latin American Symposium on Theoretical Informatics (LATIN 2006)*, volume 3887 of *Lecture Notes in Computer Science*, pages 757–768. Springer, 2006.

31. N. Pisanti, H. Soldano, M. Capentier, and J. Pothier. Implicit and explicit representation of approximated motifs. In C. Iliopoulos, K. Park, and K. Steinhofel, editors, *Algorithms for Bioinformatics*. King's College London Press, 2006.
32. N. Pisanti, H. Soldano, and M. Carpentier. Incremental inference of relational motifs with a degenerate alphabet. In *Proceedings of the 16th Annual Symposium on Combinatorial Pattern Matching (CPM 2005)*, pages 229–240, 2005.
33. E. Redhead and T. L. Bailey. Discriminative motif discovery in DNA and protein sequences using the DEME algorithm. *BMC bioinformatics*, 8(1), 2007.
34. I. Rigoutsos, A. Floratos, L. Parida, Y. Gao, and D. Platt. The emergence of pattern discovery techniques in computational biology. *Metabolic Engineering*, 2:159–177, 2000.
35. K. Romer, G. R. Kayombya, and E. Fraenkel. WebMOTIFS: automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and bayesian approaches. *Nucleic Acids Research*, June 2007.
36. M.-F. Sagot. Spelling approximate repeated or common motifs using a suffix tree. In *Proceedings of the Third Latin American Symposium: Theoretical Informatics (LATIN 1998)*, pages 374–390, 1998.
37. M. C. Shaner, I. M. Blair, and T. D. Schneider. Sequence logos: A powerful, yet simple, tool. In T. N. Mudge, V. Milutinovic, and L. Hunter, editors, *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences (HICSS 1993), Volume 1: Architecture and Biotechnology Computing*, pages 813–821, Los Alamitos, CA, 1993. IEEE Computer Society.
38. H. Soldano, A. Viari, and M. Champesme. Searching for flexible repeated patterns using a non-transitive similarity relation. *Pattern Recognition Letters*, 16(3):233–246, 1995.
39. M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Regnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137–144, 2005.
40. E. Ukkonen. Maximal and minimal representations of gapped and non-gapped motifs of a string. *Theoretical Computer Science*, 410(43):4341–4349, 2009.
41. O. V. Vishnevsky and N. A. Kolchanov. ARGO: A web system for the detection of degenerate motifs and large-scale recognition of eukaryotic promoters. *Nucleic Acids Research*, 33(Suppl.2):W417–W422, 2005.
42. G. Wang, T. Yu, and W. Zhang. WordSpy: Identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic Acids Research*, 33(Suppl.2):W412–W416, 2005.
43. E. Wijaya, S.-M. Yiu, N. T. Son, R. Kanagasabai, and W.-K. Sung. MotifVoter: A novel ensemble method for fine-grained integration of generic motif finders. *Bioinformatics*, 24:2288–2295, October 2008.
44. R. Wu, C. Chaivorapol, J. Zheng, H. Li, and S. Liang. fREDUCE: Detection of degenerate regulatory elements using correlation with expression. *BMC Bioinformatics*, 8(1):399, 2007.
45. S. Wu and U. Manber. Fast text searching: Allowing errors. *Communications of the ACM*, 35:83–91, 1992.
46. S. Zhang, W. Su, and J. Yang. ARCS-Motif: discovering correlated motifs from unaligned biological sequences. *Bioinformatics*, 25:183–189, January 2009.