

Article

Filtering Degenerate Patterns with Application to Protein Sequence Analysis

Matteo Comin ^{1,*} and Davide Verzotto ²

¹ Department of Information Engineering, University of Padova, Padova 35131, Italy

² Computational and Systems Biology, Genome Institute of Singapore, Singapore 138672, Singapore;
E-Mail: verzottod@gis.a-star.edu.sg

* Author to whom correspondence should be addressed; E-Mail: comin@dei.unipd.it;
Tel.: +39-049-827-792-8; Fax: +39-049-8277799.

Received: 29 March 2013; in revised form: 30 April 2013 / Accepted: 3 May 2013 /

Published: 22 May 2013

Abstract: In biology, the notion of degenerate pattern plays a central role for describing various phenomena. For example, protein active site patterns, like those contained in the PROSITE database, e.g., $[FY]DPC[LIM][ASG]C[ASG]$, are, in general, represented by degenerate patterns with character classes. Researchers have developed several approaches over the years to discover degenerate patterns. Although these methods have been exhaustively and successfully tested on genomes and proteins, their outcomes often far exceed the size of the original input, making the output hard to be managed and to be interpreted by refined analysis requiring manual inspection. In this paper, we discuss a characterization of degenerate patterns with character classes, without gaps, and we introduce the concept of *pattern priority* for comparing and ranking different patterns. We define the class of *underlying patterns* for filtering any set of degenerate patterns into a new set that is linear in the size of the input sequence. We present some preliminary results on the detection of subtle signals in protein families. Results show that our approach drastically reduces the number of patterns in output for a tool for protein analysis, while retaining the representative patterns.

Keywords: pattern discovery and filtering; degenerate patterns; analysis of biological data

1. Introduction

In biology, the notion of degenerate pattern, or indeterminate pattern, plays a central role for describing various phenomena. For example, protein functional patterns, like those contained in the PROSITE database [1], e.g., $[FY]DPC[LIM][ASG]C[ASG]$, are, in general, represented by degenerate patterns with character classes and denote conserved sites in protein families.

The *de novo* discovery of degenerate patterns in protein and genome sequences is very important [2–9]. Such patterns usually correspond to residues conserved by evolution, due to some significant structural or functional role. Moreover, the large availability of biological sequences, recently achieved with high-throughput sequencing technologies and a multitude of new protein discoveries, has increased the number and complexity of patterns required by scientists in order to perform a complete analysis of biological processes.

In order to fill this gap, researchers have developed several approaches over the years, following the framework of *de novo* degenerate pattern discovery [3,10–13]. Pattern discovery techniques have been used for a wide range of applications, from data compression [14,15] to data classification [16]; from protein family detection [17,18] to whole genome comparison [19], as well as the discovery of transcription factor binding sites (TFBS) [20,21]. Although *de novo* degenerate pattern discovery methods have been exhaustively and successfully tested on genomes and sets of proteins [12,18,22,23], their outcome, a huge set of patterns, often far exceeds the size of the original input, making it impractical to be managed, and then interpreted with further analysis requiring manual inspection.

In this paper, we address the problem of degenerate pattern comparison and filtering, in order to shrink the output of any pattern discovery tool for the *de novo* identification of subtle biological signals. We first give a characterization of degenerate patterns, in particular of patterns with character classes without wildcards/gaps, and define the notions of *minimality* and *pattern priority* for comparing and ranking different patterns. Then, we introduce the concept of *underlying pattern* for filtering any set of degenerate patterns into a new set that is linear in the size of the reference sequence s . Each underlying pattern will have the highest priority in some regions of s . We finally discuss some experimental results on the identification of subtle signals in protein families by means of underlying patterns. Preliminary results show that our approach drastically reduces the number of patterns in output from a *de novo* pattern discovery tool [12] for protein analysis, retaining the actual representative patterns [24].

1.1. Ranking and Clustering Degenerate Patterns

In this paper, we are interested in patterns with character classes, also called *degenerate patterns* (with no wildcards/gaps), which represent highly conserved parts in more complex signatures of protein families sharing remote homologies. The classification of their symbols often does not completely partition the entire alphabet of amino acids, and thus, in practical cases, patterns show classes with non-empty intersections. Let us consider, for example, the nickel-dependent hydrogenase large subunit signature $RG[LIVMF]E\dots\dots[QESMP][RK]C[GR][LIVM]C$. The symbols of degenerate patterns contained in this signature allow for intersections between classes, like that of $[RK]$ with $[GR]$, and also subsets, like $[LIVM]$ with respect to $[LIVMF]$.

In case the considered classes of characters form a partition of the original alphabet, Σ , we can map the input sequences into the new alphabet of symbols represented by the partition, thus considerably reducing the number of candidate patterns to be analyzed to find the real signature. Unfortunately, if we consider the (already discovered) signature reported in PROSITE, most of the time, this is not true. In most cases, this will result in the combinatorial explosion of candidate patterns to be considered [12,25], which consequently affects the efficiency of matching and discovering of new patterns.

To cope with the combinatorial explosion of degenerate patterns, *de novo* pattern discovery tools must first shrink the search space. In this regard, a number of different techniques have been proposed over the past two decades [2–12] on the basis of which other work rests [26–31]. Despite that, the number of patterns in the output still remains intractable in most cases.

Moreover, all pattern discovery tools must ultimately rank the output, according to some measure of importance. However, even if very sophisticated probabilistic scoring mechanisms are in place in a tool for protein analysis, such as Varun [12,25], there are cases where the direct interpretation of results is far from trivial. Let us consider, for instance, the worst case of Varun presented in Table 1. The relevant signatures, shown in bold, appear at position 42, whereas almost all top signatures are somehow very similar. In order to filter this output and enhance its readability, there are two main issues. On the one hand, most signatures are very similar in the contained degenerate patterns, and therefore, they must be clustered together. On the other hand, if we are interested in a specific region of the sequences under examination, we would like to select the most important signature that appears in that region according to some rule. These two issues are, in practice, tightly related and need to be addressed as one. In this article, we focus our attention on degenerate patterns that are the fundamental units of representative signatures, e.g., *short linear motifs* (SLiMs), in order to identify those representing the same loci, and thus, reducing the number of candidate solutions to be tested. In particular, we will follow and adapt the idea of pattern *minimality* introduced in [32] for exact patterns to the case of degenerate patterns. This latter notion considers exact patterns representing particular equivalent classes, or sets of locations, with the smallest possible content. In the same way, we will identify the unique degenerate pattern characterizing a set of locations with the minimal degeneracy among the patterns given in the input.

Recently, a number of ensemble methods addressing some of these clustering issues have been proposed [33–36]. The general idea is that one can integrate the outcome of several pattern discovery algorithms based on sophisticated heuristics, in order to improve the ability of finding subtle biological signals in specific contexts.

The ensemble methods actually rank all the predicted patterns according to some scoring function and then report the top patterns. WebMotifs [33], ARCS-Pattern [34], MotifMiner [35] and MotifVoter [36] assume that the consensus of several state-of-the-art tools is likely to produce the actual functional pattern. They ultimately cluster all patterns and report only those from the best clusters. However, if none of the patterns from the individual finders can accurately capture the transcription factor binding sites, the performance of the ensemble methods will suffer. Although these methods help to improve the performance of pattern finding, the improvement is usually not significant. For example, as reported in [36], the application of most ensemble tools in Tompa's benchmark [37] and *Escherichia coli* datasets improved the average sensitivity by 62%, but the average precision is reduced by 15%; only MotifVoter promises to improve these numbers.

Table 1. Output of Varun for the G-protein coupled receptors family 3 (id PS00980), consisting of 25 sequences of about 25,000 amino acids each. One of the relevant signatures is shown in bold [12].

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

1.2. Problem Formulation

More formally, the problem we address is the following: Given a reference sequence, s , which is the concatenation of multiple protein sequences and a set of patterns with character classes, \mathcal{M} , which is the

outcome of one or more pattern finders, reduce the set, \mathcal{M} , to a small number of ordered representative patterns, say \mathcal{U} , such that every position of s is covered by at most one pattern of \mathcal{U} .

In this paper, we will discuss the basic problems behind degenerate pattern comparison and ranking, in a simple and conservative fashion; in particular, we will focus our attention on protein sequences with remote evolutionary relationships. In principle, there are several ways to compare degenerate patterns, and they use the number of occurrences or the pattern locations; we will consider only the pattern locations. We propose a simple, yet effective, way that will be compared with other traditional binary relations in Section 5. There are mainly three features that characterize a degenerate pattern: Length, degeneracy and its location list. A trade-off between these three features is required in order to compare and rank all patterns [38]. We chose the combination, called *pattern priority*, which tries to represent most of the representative patterns present in PROSITE. An intuitive way to rank all patterns will be to favor degenerate patterns representing large matching regions between sequences and that are more likely to be “functional” [39], and thereafter, those patterns with the least degeneracy, which represents the degree of divergence during evolution of sequences, and it is “a prerequisite for and an inescapable product of the process of natural selection itself” [40].

Finally, here, our main intent is to use and experimentally validate our heuristic pattern priority rule. This can be viewed as the first step towards an efficient *de novo* pattern discovery algorithm able to directly discover conserved patterns, without computing a (possible) exponential number of degenerate patterns. In fact, it is well known that the problem of finding degenerate patterns consistent with a set of heterogeneous sequences is, in general, NP-complete [41], and thus, fast heuristics are required.

2. Preliminary Definitions

A *string* is a sequence of symbols from an alphabet, Σ . The set of all strings over Σ is denoted by Σ^* . The length of a string, s , is denoted by $|s|$ and the i -th symbol of s is s_i , where $1 \leq i \leq |s|$. Let, now, $s = s_1s_2 \dots s_n$ be a string of length $|s| = n$ on Σ . This is the reference sequence and, in practice, will be the concatenation of several sequences. For ease of explanation, here, we consider only one sequence, and in Section 5, we will clarify how to deal with multiple sequences. A symbol from Σ , say σ , is called a *solid character*. A character class, C , is a subset of Σ of cardinality of at least two. Let us assume that we have several character classes, C_j . For the rest of the paper, we assume that we are given a string, s , on Σ of length, n , and a positive integer, q , $2 \leq q \leq |s|$, called *quorum*.

Definition 1 (*Sequence*) A sequence with character classes, or simply a sequence, is a string of consecutive symbols defined on $\{2^{C_j}\}$.

In the literature, a sequence with character classes is also called a *degenerate pattern* or an *indeterminate string* [4–9]. Let p be a sequence with character classes; then, we write its symbols by means of square brackets. For example, if $p = p_1p_2 \dots p_k$, where $|p| = k$, and each symbol, p_j , with $1 \leq j \leq k$, is either a solid character or a character class, C , and is written as $p_j = [\sigma_1, \sigma_2, \dots, \sigma_{r_j}]$. As of Table 2, $s = aabeadbace$, $\Sigma = \{a, b, c, d, e\}$, $q = 3$, and the two classes are $C_1 = \{a, c, d\}$ and $C_2 = \{a, b, e\}$, then $p = a[a, c, d][b, e]$ is a sequence with character classes of length $k = 3$.

Table 2. Example of pattern. The occurrences of p in s are drawn in black.

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

j	1	2	3
p_j	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
$occurrences$											

The main issue with this example is that the character classes might not be a partition of Σ . This is very common for PROSITE functional patterns, e.g., $[LIVMA][LIVMY].G[GSTA][DES]L[FI][TN][GS]$.

Definition 2 (Sequence occurrence) A sequence, $p = p_1p_2 \dots p_k$, 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 $1 \leq j \leq k$.

A pattern, m , is then defined as a pair, (p, \mathcal{L}_m) , where p is a sequence 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, in Table 2, $m = (p, \mathcal{L}_m)$ is a pattern with sequence $p = a[a, c, d][b, e]$ of length $k = 3$ and location list $\mathcal{L}_m = \{1, 5, 8\}$. More formally:

Definition 3 (Pattern, location list) We say that $m = (p, \mathcal{L}_m)$ is a pattern with sequence, $p = p_1p_2 \dots p_k$, and location list, $\mathcal{L}_m = (l_1, l_2, \dots, l_\nu)$, if and only if all of the following hold: (i) p has at least two symbols, $|p| \geq 2$; (ii) The location list has at least q occurrences, $|\mathcal{L}_m| \geq q$; and (iii) There does not exist a location, $l' \notin \mathcal{L}_m$, such that p occurs at l' in s (that is, \mathcal{L}_m is complete).

3. Minimal Patterns and Pattern Priority

In this section, we introduce the notions of *minimality* and *pattern priority*. The former will be used to avoid useless characters in the definition of a pattern, the latter as a means for comparison.

Definition 4 (Minimal representation $\mu(\cdot)$) Given a sequence, p , of length, k , the minimal representation of p is a sequence, $\mu(p)$, of length, k , with symbols, $\mu(p)_j = \bigcup_{l \in \mathcal{L}_m} s_{l+j-1}$, for $1 \leq j \leq k$.

Remark 1 The minimal representation of a sequence is unique.

Remark 2 Since $\mu(p)$ is more specific than p , that is, $\mu(p)$ cannot have more occurrences in s than p , then the list of occurrences of $\mu(p)$ must be the same as p .

Let $\mu(m) = (\mu(p), \mathcal{L}_m)$ be the minimal representation of a pattern, m , with sequence, p , then Remark 2 suggests to us that $\mu(m)$ agrees with the definition of pattern (that is, $\mathcal{L}_{\mu(m)}$ is complete):

Definition 5 (Minimal pattern) *The minimal representation of m , given by $\mu(m) = (\mu(p), \mathcal{L}_m)$, is called a minimal pattern.*

Computing the minimal representation of a pattern is useful when a pattern is composed by character classes. To have a more concrete idea about this concept, Table 3 shows an example of minimal representation of the pattern $m = (p = [a, c, d][a, c, d][a, b, e], \{1, 5, 8\})$, where the reference string is $s = aabeadbace$ and the classes are $C_1 = \{a, c, d\}$ and $C_2 = \{a, b, e\}$. In this case, we say that $\mu(m) = (\mu(p) = a[a, c, d][b, e], \{1, 5, 8\})$ is a minimal pattern. In practice, most of the functional patterns reported by PROSITE are not minimal.

Table 3. Example of minimal representation of a pattern, m , with sequence, p .

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

Let \mathcal{M} be a set of patterns with character classes lying on the string, s . From Remark 1 we have that each pattern, $m \in \mathcal{M}$, has a unique minimal representation, $\mu(m)$. Thus, one can easily check that $\mu(m) = \mu(m')$ is an equivalence relation. Let us map all the patterns in \mathcal{M} into the set of their minimal version, $\mu(\mathcal{M})$, where each pattern, $m \in \mu(\mathcal{M})$, is the minimal, $m = \mu(m')$, of some pattern, $m' \in \mathcal{M}$. Then, the set of patterns, \mathcal{M} , is partitioned into equivalence classes by the binary relation of equality between minimal patterns.

Remark 3 *Two patterns, m and m' , in \mathcal{M} may have the same location list; thus, they will be mapped into the same minimal pattern, $\mu(m) = \mu(m')$. On the other hand, two minimal patterns with the same location list must have different lengths.*

We call $\mu(\mathcal{M})$ the *minimal set* of \mathcal{M} . Since mapping \mathcal{M} in $\mu(\mathcal{M})$ could mean a drastic reduction in the number of patterns, this is, in practice, a first step in filtering. Now, we define a simple property of patterns with character classes, the *degeneracy*, that is, the number of characters in a pattern.

Definition 6 (Degeneracy of a sequence $c(\cdot)$) *The degeneracy of a sequence, p , of length, k , is defined as $c(p) = \sum_{j=1}^k |p_j|$. The degeneracy of a pattern, $m = (p, \mathcal{L}_m)$, denoted by $c(m)$, is defined as the degeneracy of its sequence, $c(p)$.*

For instance, given two sequences, $p = a[a, c, d][b, e]$ and $p' = [a, d]b[a, b]$, their degeneracy is $c(p) = 6$ and $c(p') = 5$. Therefore, the degeneracy of the pattern, $m = (a[a, c, d][b, e], \{1, 5, 8\})$, is equal to $c(p)$, that is, 6.

With this notion, we can define the *priority* between patterns, as a means for comparing different patterns. Note that several notions of priority can be established at this stage. We choose a very intuitive combination of pattern length, pattern degeneracy and location list.

Definition 7 (Pattern priority ' \rightarrow ') *A pattern, m , of length, k , has priority over another pattern, m' , of length, k' , denoted, $m \rightarrow m'$, if (1) $k > k'$, or (2) $k = k'$ and $c(m) < c(m')$, or (3) $k = k'$, $c(m) = c(m')$, $\min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{\mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, 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 they have the same length and degeneracy, but $1 = \min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{\mathcal{L}_{m'} \setminus \mathcal{L}_m\} = 2$. If two patterns, m and m' , have the same length and degeneracy, but either $\mathcal{L}_m \subseteq \mathcal{L}_{m'}$ or $\mathcal{L}_{m'} \subset \mathcal{L}_m$, then we consider them *incomparable*; otherwise, they are comparable. Nevertheless, we will solve the non-comparability of two patterns with Theorem 2.

Given our binary relation, called *pattern priority*, we want to prove that some basic properties holds:

Definition 8 (*Sub-ordered, totally ordered*) Given a set, \mathcal{M} , if a binary relation, R , over this set is (i) *irreflexive*; $m R m'$ never holds (ii) *antisymmetric*, $m \neq m' \Rightarrow \text{not } (m R m' \text{ and } m' R m)$; and (iii) *acyclic*, then \mathcal{M} is said to be *sub-ordered*. If R is also *total*, that is $(m R m' \text{ or } m' R m)$, then \mathcal{M} is said to be *totally sub-ordered*. Furthermore, \mathcal{M} is said to be *totally ordered* if R is *irreflexive*, *antisymmetric*, *transitive*, that is $(m R m' \text{ and } m' R m'') \Rightarrow m R m''$ and *total*.

Lemma 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 implies acyclicity, that is, if \mathcal{M} is *totally ordered*, then \mathcal{M} is also *totally sub-ordered*. It is also easy to see that acyclicity and totality, together, imply transitivity. Consider a chain of patterns in \mathcal{M} : $m_1 R m_2, m_2 R m_3, \dots, m_{t-1} R m_t$. Since for acyclicity, $m_t R m_1$ does not hold, then, for totality, $m_1 R m_t$. Hence, transitivity holds and, therefore, the definitions *totally sub-ordered* and *totally ordered* coincide. \square

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

Fact 1 The binary relation of pattern priority is *irreflexive* and *antisymmetric* (since properties (1), (2) and (3) of Definition 7 are strictly defined).

Lemma 2 Let m and m' be two patterns with $\min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\} < \min\{\mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, such that the two minima both exist, and define j to be $\min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\}$. Then, the occurrences of m and m' at positions less than j must be the same.

Proof We have to prove that, in case (3) of Definition 7, 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 $\min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\} < j$, which is impossible by hypothesis. Conversely, if m' has occurrences less than j that are not in \mathcal{L}_m , then $\min\{\mathcal{L}_{m'} \setminus \mathcal{L}_m\} < j = \min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ contradicts again our assumptions. Thus, the occurrences of the two patterns less than j must be the same, and we call them *paired*. Finally, by assumptions, it trivially holds that $j \notin \mathcal{L}_{m'}$. \square

Lemma 3 Let $m_1 \rightarrow m_2, m_2 \rightarrow m_3, \dots, m_{t-1} \rightarrow m_t$ be a chain of patterns with the same length and degeneracy. 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 will show now that, if it holds either $m_1 \rightarrow m_t$ or $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$, then either $m_1 \rightarrow m_{t+1}$ or $\mathcal{L}_{m_{t+1}} \subset \mathcal{L}_{m_1}$ holds.

Assume $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$. Define j to be $\min\{\mathcal{L}_{m_t} \setminus \mathcal{L}_{m_{t+1}}\}$, this minimum exists, since $m_{t-1} \rightarrow m_t$ (by property (3) of Definition 7). It follows that $j \in \mathcal{L}_{m_1}$ and, from Lemma 2, that the occurrences of m_t and m_{t+1} that are less than j are paired. Hence, these occurrences are shared also with m_1 . Since $j \notin \mathcal{L}_{m_{t+1}}$, then either $m_1 \rightarrow m_{t+1}$ holds (because j makes the difference between \mathcal{L}_{m_1} and $\mathcal{L}_{m_{t+1}}$) or m_1 and m_{t+1} are incomparable. In the latter case, $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_1}$ and $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_{t+1}}$ imply $\mathcal{L}_{m_t} \subset \mathcal{L}_{m_{t+1}}$, which is impossible, since m_t and m_{t+1} are comparable by hypothesis. Thus, in this latter case, we have that $\mathcal{L}_{m_{t+1}} \subset \mathcal{L}_{m_1}$.

Conversely, assume $m_1 \rightarrow m_t$. Define j as before and $j' = \min\{\mathcal{L}_{m_1} \setminus \mathcal{L}_{m_t}\}$. It holds that $j \notin \mathcal{L}_{m_{t+1}}$ (as already observed), and that $j' \neq j$, because of $j' \notin \mathcal{L}_{m_t}$. Then, by Lemma 2, the occurrences of m_t and m_{t+1} that are less than j are paired, and the same is true for the occurrences of m_1 and m_t that are less than j' . It follows that, if m_1 and m_{t+1} are comparable, there exists an occurrence in \mathcal{L}_{m_1} that is more on the left with respect to those of $\mathcal{L}_{m_{t+1}}$: If $j' < j$, the occurrences of m_1, m_t and m_{t+1} less than j' are paired together, and thus, j' makes the difference; otherwise ($j' > j$), the occurrences of m_1, m_t and m_{t+1} less than j are paired together, and hence, $j \in \mathcal{L}_1$ makes the difference. Alternatively, $\mathcal{L}_{m_1} \subseteq \mathcal{L}_{m_{t+1}}$ implies that the occurrences of m_1 equal to or less than j' are shared with m_{t+1} , which leads to $m_{t+1} \rightarrow m_t$, which is impossible. \square

Theorem 1 Any set of patterns, \mathcal{M} , is sub-ordered with respect to the binary relation of pattern priority.

Proof We have to prove that the relation of pattern priority is irreflexive, antisymmetric and acyclic. The first two properties are stated in Fact 1. Now, following the work in Lemma 3, we can prove that the acyclicity holds too. First, observe that length and degeneracy are intrinsic properties of the single pattern. If all patterns in \mathcal{M} have different lengths and degeneracies, then by definition of pattern priority, it is always true that either $m \rightarrow m'$ or $m' \rightarrow m$, and a cycle can never exist, because of different lengths or degeneracies. Alternatively, consider a chain of patterns, $m_1 \rightarrow m_2, \dots, m_{t-1} \rightarrow m_t$, with the same length and degeneracy. In this case, we must use property (3) of Definition 7 to compare the patterns together. From Lemma 3, it follows that a cycle of pattern priority between any chain of patterns is, again, impossible, and hence, the acyclicity holds. \square

Note that the non-decision on some binary comparisons finally discards the relation of totality and, as seen above, of transitivity. For instance, consider $s = \text{aabeadbace}$ and the patterns $m = ([a, d] [a, b] [a, e], \{2, 6\})$ 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 pattern 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 these patterns are not minimal, like most of the patterns in PROSITE. In the following, we set the basis to solve this problem.

Theorem 2 Given any set of patterns, \mathcal{M} , its minimal set, $\mu(\mathcal{M})$, is totally ordered under the binary relation of pattern priority.

Proof Along with Theorem 1, we have that any set of patterns, \mathcal{M} , is sub-ordered under pattern priority; thus, also the set of minimal patterns, $\mu(\mathcal{M})$, is sub-ordered. Following Lemma 1, we have to prove that the totality holds on this new set, $\mu(\mathcal{M})$, that is, every pair of minimal patterns must be comparable under pattern priority. In other words, if $m \neq m'$, it must hold either $m \rightarrow m'$ or $m' \rightarrow m$.

From Remark 3, we have that $\mathcal{L}_m \neq \mathcal{L}_{m'}$ for two minimal patterns, m and m' , with the same length, and thus, we have, without loss of generality, two cases to consider: $\mathcal{L}_m \not\subseteq \mathcal{L}_{m'}$ or $\mathcal{L}_m \subset \mathcal{L}_{m'}$. From now, m and m' will be two minimal patterns with the same length and, for the former case, also with the same degeneracy $c(m)$. Thus, in the former case, if we consider $\min\{\mathcal{L}_m \setminus \mathcal{L}_{m'}\}$ and $\min\{\mathcal{L}_{m'} \setminus \mathcal{L}_m\}$, the two minima exist and are different from each other; hence, it holds that either $m \rightarrow m'$ or $m' \rightarrow m$, respectively, if the minimum of the two sets is either in \mathcal{L}_m or in $\mathcal{L}_{m'}$. In the latter case, $\mathcal{L}_m \subset \mathcal{L}_{m'} \Rightarrow c(m) < c(m')$, since m and m' are minimal patterns, and the respective location lists are complete (that is, the respective patterns of m and m' must be different); therefore, it holds that $m \rightarrow m'$. Thus, for any set of minimal patterns under pattern priority, the totality holds. From Lemma 1, we can conclude that any set of minimal patterns is totally ordered under the pattern priority rule. \square

As a consequence of Theorem 2, all minimal patterns can be compared and ranked. We can further observe that every minimal pattern has priority over the patterns within its equivalence class, due to property (2) of pattern priority. Now, it is clear that any set of patterns can be mapped into its minimal representative set and that we can build a measure of total order over this set.

4. Pattern Filtering

Here, we describe an application of pattern priority; the objective is to select the most important patterns in $\mu(\mathcal{M})$ for each location of s , according to our pattern priority rule. If a pattern, m , is selected, we filter out all patterns with less priority that lay on the same locations of m . If these locations are, for example, transcription factor binding sites or coding sequences of a genome, we will select only one pattern that best represents these locations. We say that an occurrence l of m is tied to an occurrence l' of m' , if the two occurrences overlap, *i.e.*, $([l, l + |m| - 1] \cap [l', l' + |m'| - 1]) \neq \emptyset$. Otherwise, we say that l is untied from l' .

Definition 9 (*Underlying pattern*) *The set of patterns, $\mathcal{U} \subseteq \mu(\mathcal{M})$, is said to be underlying if and only if:*

- (i) *Every pattern, m , in \mathcal{U} , called an underlying pattern, has at least q occurrences that are untied from all the untied occurrences of other patterns in $\mathcal{U} \setminus m$ and*
- (ii) *There does not exist a pattern, $m \in \mu(\mathcal{M}) \setminus \mathcal{U}$, such that m has at least q untied occurrences from all the untied occurrences of patterns in \mathcal{U} .*

This subset of $\mu(\mathcal{M})$ is composed only by those patterns that rank higher in our priority rule for some location of s . The following algorithm filters any set of patterns, \mathcal{M} , into the reduced set of underlying patterns, \mathcal{U} .

Underlying Pattern Filtering (Input: \mathcal{M} , q ; Output: \mathcal{U})

1. Compute the minimal set, $\mu(\mathcal{M})$.

2. Rank all minimal patterns in $\mu(\mathcal{M})$ using the pattern priority rule.
3. At each step, select the top pattern, m , from $\mu(\mathcal{M})$:
 - If all of its occurrences are tied/covered by some other patterns already in \mathcal{U} , discard m ;
 - Otherwise, if m has at least q untied occurrences, add m to \mathcal{U} and update the locations of vector, Γ , in which m appears.

The correctness of the algorithm follows from Theorem 2. Let n be the size of the string, s . During the first step, in the worst case, a pattern, m , can have a location list, $|\mathcal{L}_m|$, that is $O(n)$, and the comparison of two ordered lists of occurrences costs $O(n)$. Thus, this step of the algorithm costs $O(n^2|\mathcal{M}|)$. The second phase orders the set of $\mu(\mathcal{M})$, where each comparison between two patterns can take $O(n)$ times; thus, this step costs $O(n|\mu(\mathcal{M})| \log |\mu(\mathcal{M})|)$.

In the third step, for each pattern, m , that has been selected by our algorithm, we store the occurrences of m in a vector of Booleans Γ , that represents the locations of s . This means that, $\forall l \in \mathcal{L}_m$, we store the value TRUE in the locations $\Gamma[l + i]$, for $0 \leq i < |m|$. This vector is then used to check if an occurrence is untied in constant time. Thus, for each pattern in $\mu(\mathcal{M})$, checking for untied occurrences and updating the vector, Γ , takes $O(n)$ time. In total, step 3 requires $O(n|\mu(\mathcal{M})|)$ time.

In short, the total complexity of the algorithm is $O(n^2|\mathcal{M}|)$. Note that this complexity does not depend on the notion of pattern priority used; thus, also, other binary relations can be applied with the same complexity. If the structure of the pattern priority is considered, it is possible to reduce the complexity using some results developed for patterns without character classes [19]. However, this is out of scope for this paper, since the Underlying Pattern Filtering is very fast in practice.

Finally, we observe that the untied occurrences of all patterns, m , in \mathcal{U} are non-overlapping. From this consideration, we have the following result:

Corollary 1 *The number of patterns in \mathcal{U} is $\leq \lceil n/2 \rceil$, independently of the size of \mathcal{M} .*

As a consequence, the set, \mathcal{U} , has linear dimension with respect to the size of s .

5. Experimental Results

In this section, we discuss the ability of underlying patterns to efficiently capture meaningful biological information. A general problem in genome and proteome research is the identification of signals represented by means of degenerate patterns. In this sense, some modern degenerate pattern discovery tools proved to be useful in biological sequence analysis, as we have discussed at the beginning of the paper. Here, we first collect the degenerate patterns in the output from one of these tools, say a set of patterns, \mathcal{M} , and then, present some preliminary experimental results in order to support the theoretical properties shown in the above sections.

More generally, there are two types of scenarios where the notion of underlying patterns could 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 patterns that are underlying with respect to that particular region, without considering the whole set of patterns. Another possible application is the case where we just want to filter all patterns 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 patterns extracted by Varun [12,25], a tool for *de novo* pattern discovery. The dataset consists of six protein families for which Varun successfully extracts the representative patterns contained in the PROSITE signatures (release 20.85) [12]. For each signature, we select all sequences in the Swiss-Prot database that share that signature. In summary, our dataset is the following.

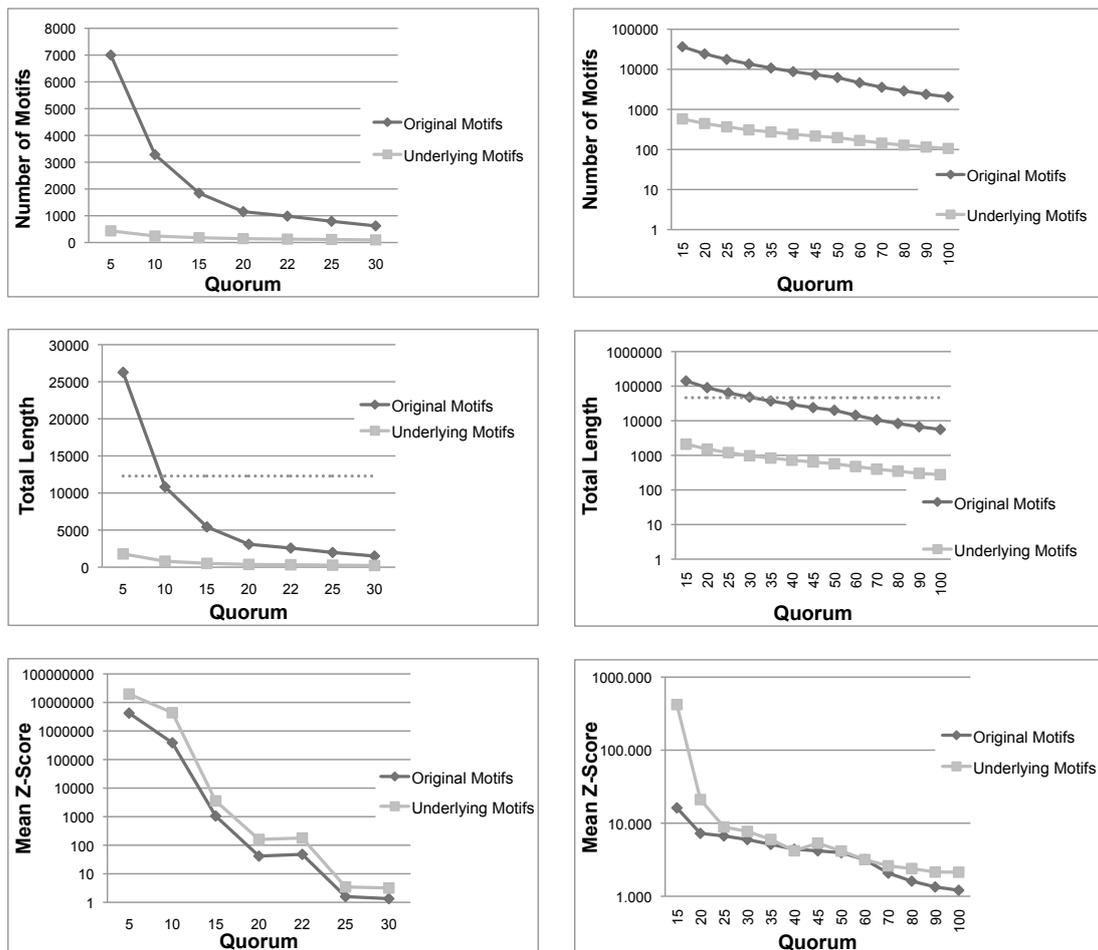
1. Nickel-Dependent hydrogenases (id PS00508; in short, *Ni*). These are enzymes that catalyze the reversible activation of hydrogen and are further involved in the binding of nickel. The family is composed by 22 sequences of about 12,300 amino acids in total. This family contains two representative signatures, $Ni_1 = RG[FILMV]E.....[EM PQS][KR].C[GR][ILMV]C$ and $Ni_2 = [FY]D[IP][CU][AILMV][AGS]C$.
2. Coagulation factors 5/8 type C domain (FA58C) (id PS01286; in short, *Fa*). This family is composed by 40 sequences of about 46,500 amino acids in total. They share two signatures: $Fa_1 = [FWY][ILV].[AFILV][DEGNST].....[FILV]..[IV].[ILTV][KMQT]G$ and $Fa_2 = [LM]R.[EG][ILPV].GC$.
3. Formate and nitrite transporters (id PS01005; in short, *Form*). The signature $[LIVMA][LIVMY].G[GSTA][DES]L[FI][TN][GS]$ is present in 17 sequences of a total length of 5300 amino acids.
4. Ubiquitin-Activating enzyme (id PS00865; in short, *Ubi*). The active site $P[LIVMG]CT[LIVM][KRHA].[FTNM]P$ appears in 36 proteins of about 25,200 amino acids in total.
5. RNA polymerases M/15 Kd subunits (id PS01030; in short, *Poly*). The representative signature $[FY]C.[DEKSTG]C[GNK][DNSA][LIVMHG][LIVM]$ occurs in 29 sequences of about 4000 amino acids.
6. Dbl homology domain (id PS00741; in short, *Dbl*). The signature $[LM]..[LIVMFYWGS][LI]..[PEQ][LIVMRF]..[LIVM].[KRS].[LT].[LIVM].[DEQN][LIVM]..[STM]$ appears in 65 sequences of a total length of 18,750 amino acids.

To prove that the information captured by the underlying patterns is relevant with respect to the protein families under examination, we devised two kinds of tests. In the first test, we compare the original set of patterns in the input, \mathcal{M} , with the set of underlying patterns in the output, \mathcal{U} , using global measures. In the second, we investigate the ability to filter a list of meaningful patterns and retain the representative ones, perhaps with a higher rank.

In the first set of experiments, we use Varun to extract patterns from the above families of protein sequences for different quorums, q . For each family, we concatenate the sequences and extract patterns from the concatenation; thus, in all the experiments, the quorum refers to the number of occurrences in the concatenation of all sequences of a certain family. This notion of quorum can generate several spurious patterns and, thus, will make the filtering problem more challenging. We also investigate a different setup in which the quorum value reflects the number of sequences where the pattern is contained. This latter scenario produces similar results; however, the range of values for this quorum is bounded by the number of sequences, and this can limit the possibility to study the efficiency of the filtering process, while varying the quorum values. For these reasons, we choose to use the first definition

of quorum. Then, we use the extracted patterns as input to our algorithm, presented in Section 4, in order to compute the underlying patterns \mathcal{U} . In our experiments, each pattern, $m \in \mathcal{M}$, extracted by Varun is with character classes and no gaps, where we set the quorum, q , for the underlying patterns, to be the same as the quorum for the original patterns in the input, as by the definition seen above. The length, n , of the string in the input is equal to the length of the concatenations reported above. The size of \mathcal{M} varies with respect to the quorum, q , and for some configurations, can be larger than the sequence length (see Figure 1). All experiments were conducted on a common PC, and the filtering process requires on average less than 10 s.

Figure 1. Total number, sum of lengths and mean z -score of the patterns extracted using Varun and their corresponding underlying patterns, for the two protein families, Ni and Fa . The dashed line in the total length diagrams indicates the total size of each family. Note that in (a) Mean Z-Score and (b) All diagrams, the ordinate is plotted on a logarithmic scale. (a) Nickel-Dependent hydrogenases (Ni); (b) Coagulation factors 5/8 type C domain (Fa).



For both sets of patterns, \mathcal{M} and \mathcal{U} , we compute some global statistics. Figure 1 shows the number of patterns, the sum of lengths of all patterns and their mean z -score for the first two protein families (Ni , Fa). The other families share the same behavior (figure not shown). The z -score is computed employing the same formula reported in [12]. As expected, the number of underlying patterns is always much smaller than the number of the original patterns. A similar conclusion can be drawn for the

sum of lengths. More importantly, for small quorums, the total length of original patterns exceeds the length of sequences in both families, indicating that the original patterns are even larger than the set of sequences under examination. Moreover, as seen above, the sum of lengths of underlying patterns 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 patterns are much smaller than those of the original patterns. Therefore, the filtering process is space-efficient, as expected.

Another important measure is the mean z -score of \mathcal{M} and \mathcal{U} . The z -score of a pattern, p , is a statistical measurement of the degree of over-representation of p with respect to the expected number of its occurrences. The mean z -score is thus a global measure able to capture the average quality of patterns in a set; however, it is much more computationally expensive than our pattern priority, and we just use it to validate our approach. In Figure 1, we can see that, for all quorums, the average z -score of underlying patterns is always greater than those of original patterns, and in most cases, the difference is one or two orders of magnitude. To summarize, this first test confirms that the number and span of underlying patterns 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 patterns. To this end, we employ the candidate patterns obtained from the previous experiments to test the presence of PROSITE signatures. We consider in detail again the first two families, Fa and Ni , and the corresponding signatures. The other four families show similar results and are summarized in Table 4.

We consider Ni_2 and Fa_2 directly as degenerate patterns, due to their low number of gaps, while we split signatures, Ni_1 and Fa_1 , into two different patterns each and compute the statistics of these patterns accordingly. For both sets of patterns, \mathcal{M} and \mathcal{U} , we compute the maximum similarity between each pattern in the set and the two signatures. The similarity between two patterns, m and m' , is the number of shared characters, including character classes, in the best alignment of m versus m' , without considering indels. Tables 5 and 6 summarize the maximum similarity, for different quorums, of \mathcal{M} and \mathcal{U} , with each signature of the two protein families, Ni and Fa . The second and third columns report the maximum similarity for the set of underlying patterns divided by the similarity of the original set. For example, in the first row of Table 5, the quorum is five. In this case, the maximum similarity of \mathcal{M} with the representative pattern, Ni_1 , is 26. The same value is obtained also for the corresponding set of underlying patterns, \mathcal{U} , thus indicating that the pattern, Ni_1 , is retained with the same degree of accuracy.

Table 4. Comparison of performance between different binary relations applied to the underlying patterns: Pattern priority, z-score, probability with distribution based on the amino acid frequencies in *s*, probability with no background (*i.e.*, each amino acid scores 1/20), frequency of patterns in *s*, inverse frequency and the lexicographic order of occurrences. For each family, we summed up the maximum similarity with the two representative patterns for all quorums. Similarly, in the column rank, we show the average rank of the closest candidates to these patterns.

Binary Relation	<i>Ni</i> _{1,2}		<i>Fa</i> _{1,2}	
	Similarity	Rank	Similarity	Rank
Pattern priority	151/157	2.78	247/264	5.34
z-Score	127/157	5.00	223/264	9.96
Probability	127/157	5.00	223/264	9.96
Equal Probability	127/157	5.00	223/264	9.96
Frequency	93/157	22.78	168/264	9.42
Inverted frequency	118/157	6.14	212/264	5.69
Lexicographic order	93/157	5.50	142/264	11.77

Binary Relation	<i>Form</i>		<i>Ubi</i>		<i>Poly</i>		<i>Dbl</i>	
	Similarity	Rank	Similarity	Rank	Similarity	Rank	Similarity	Rank
Pattern priority	186/205	4.72	190/198	3.40	215/234	4.25	498/522	5.20
z-Score	167/205	6.00	178/198	4.74	212/234	5.86	455/522	7.10
Probability	167/205	6.00	178/198	4.74	210/234	5.92	455/522	7.10
Equal Probability	165/205	6.20	178/198	4.74	210/234	5.92	452/522	7.10
Frequency	102/205	26.62	112/198	9.75	135/234	13.69	321/522	21.35
Inverted frequency	154/205	7.92	159/198	6.00	188/234	10.10	436/522	13.74
Lexicographic order	105/205	16.44	112/198	12.39	126/234	13.93	308/522	22.00

Table 5. Normalized maximum similarity with the reference patterns of the family nickel-dependent hydrogenases, for different quorums.

Quorum	Max Similarity <i>Ni</i> ₁ (underlying/original)	Max Similarity <i>Ni</i> 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 6. Normalized maximum similarity with the reference patterns of the family coagulation factors 5/8 type C domain, for different quorums.

Quorum	Max Similarity Fa_1 (underlying/original)	Max Similarity 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

In principle, the binary relation of pattern priority can be replaced by any other traditional means of comparison. To this end, we compared the pattern priority rule with other standard ranking methods that were applied to the underlying filtering step. Table 4 reports the average scores for each measure for all six protein families, where a large maximum similarity with the two PROSITE signatures and a higher rank are preferable. In this context, we consider different ranking methods: The lexicographic order of patterns (Lexicographic), the number of occurrences (Frequency) and its inverse (Inverted Frequency). Other statistical ranking methods are also considered here: Pattern probability assuming either an i.i.d. distribution of symbols based on amino acid frequencies (Probability) or an equal distribution of symbols (Equal Probability); z -score, computed using the probability value as in [12].

We can easily see that our pattern priority achieves the best scores among all methods for the detection of representative patterns. In addition, our heuristic ranks on average the reference patterns of Ni in the top three out of 2239 candidate patterns in the input and those of Fa in the top five out of 10,842 patterns, while for the other families, the reference patterns range from the top three to the top five selected patterns, on average. The definition of pattern priority was conceived especially for degenerate patterns, like those presented in this section. However, this framework can be used in conjunction with other comparison functions designed specifically for patterns with profiles or variable gaps, e.g., [38].

These preliminary experiments support the validity of the theoretical results presented in the previous sections and also prove their effectiveness for protein analysis. However, a more comprehensive experimental setting is desirable, in order to improve the evidence on the performance and to compare this method with other existing tools, like MEME [42].

6. Conclusions

In this paper, we have studied patterns with character classes, introducing basic properties for the comparison and ranking of patterns. We have proven theoretical results that support the validity of these properties. Most importantly, the pattern priority rule, together with the notion of underlying patterns, has proven to be valuable for the analysis of biological sequences, bounding the total length of degenerate patterns in the output from any modern pattern discovery tool. Preliminary experiments on protein families have shown the good performance of our approach as a filter to reduce the number of patterns in the output, while keeping the representative ones. More experiments should be conducted in order to compare these results with other tools, like MEME [42]. We are currently working to extend the notion of underlying patterns for whole genome comparison.

Acknowledgments

M.C. was partially supported by the Ateneo Project CPDA110239.

References

1. Hulo, N.; Bairoch, A.; Bulliard, V.; Cerutti, L.; Cuče, B.; de Castro, E.; Lachaize, C.; Langendijk-Genevaux, P.; Sigrist, C. The 20 years of PROSITE. *Nucleic Acids Res.* **2008**, *36*, D245–D249.
2. Parida, L. *Pattern Discovery in Bioinformatics: Theory and Algorithms*; Mathematical and Computational Biology, Chapman and Hall/CRC: Boca Raton, FL, USA, 2007.
3. Jensen, K.L.; Styczynski, M.P.; Rigoutsos, I.; Stephanopoulos, G.N. A generic motif discovery algorithm for sequential data. *Bioinformatics* **2006**, *22*, 21–28.
4. Abrahamson, K. Generalized string matching. *SIAM J. Comput.* **1987**, *16*, 1039–1051.
5. Navarro, G.; Raffinot, M. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. *J. Comput. Biol.* **2003**, *10*, 903–923.
6. Fredriksson, K.; Grabowski, S. Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. *Inf. Retr.* **2008**, *11*, 335–357.
7. Wu, S.; Manber, U. Fast text searching: Allowing errors. *Commun. ACM* **1992**, *35*, 83–91.
8. Soldano, H.; Viari, A.; Champesme, M. Searching for flexible repeated patterns using a non-transitive similarity relation. *Pattern Recognit. Lett.* **1995**, *16*, 233–246.
9. Pisanti, N.; Soldano, H.; Carpentier, M. Incremental inference of relational motifs with a degenerate Alphabet. *Lect. Notes Comput. Sci.* **2005**, *3537*, 229–240.
10. Frith, M.C.; Saunders, N.F.W.; Kobe, B.; Bailey, T.L. Discovering sequence motifs with arbitrary insertions and deletions. *PLoS Comput. Biol.* **2008**, *4*, doi:10.1371/journal.pcbi.1000071.
11. Sinha, S.; Tompa, M. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res.* **2002**, *30*, 5549–5560.
12. Apostolico, A.; Comin, M.; Parida, L. VARUN: Discovering extensible motifs under saturation constraints. *IEEE/ACM Trans. Comput. Biol. Bioinforma.* **2010**, *7*, 752–762.

13. Pisanti, N.; Crochemore, M.; Grossi, R.; Sagot, M.F. Bases of motifs for generating repeated patterns with wild cards. *IEEE/ACM Trans. Comput. Biol. Bioinforma.* **2005**, *2*, 40–50.
14. Apostolico, A.; Comin, M.; Parida, L. Bridging lossy and lossless compression by motif pattern discovery. *Lect. Notes Comput. Sci.* **2006**, *4123*, 793–813.
15. Apostolico, A.; Comin, M.; Parida, L. Motifs in Ziv-Lempel-Welch Clef. In Proceedings of IEEE DCC Data Compression Conference, Snowbird, UT, USA, 23–25 March 2004; pp. 72–81.
16. Apostolico, A.; Comin, M.; Parida, L. Mining, compressing and classifying with extensible motifs. *Algorithms Mol. Biol.* **2006**, *1*, doi:10.1186/1748-7188-1-4.
17. Comin, M.; Verzotto, D. The Irredundant Class method for remote homology detection of protein sequences. *J. Comput. Biol.* **2011**, *18*, 1819–1829.
18. Comin, M.; Verzotto, D. Classification of protein sequences by means of irredundant patterns. *BMC Bioinforma.* **2010**, *11*, doi:10.1186/1471-2105-11-S1-S16.
19. Comin, M.; Verzotto, D. Alignment-Free phylogeny of whole genomes using underlying subwords. *BMC Algorithms Mol. Biol.* **2012**, *7*, doi:10.1186/1748-7188-7-34.
20. Comin, M.; Parida, L. Detection of subtle variations as consensus motifs. *Theory Comput. Sci.* **2008**, *395*, 158–170.
21. Comin, M.; Parida, L. Subtle Motif Discovery for Detection of Dna Regulatory Sites. In Proceedings of the 5th Asia-Pacific Bioinformatics Conference, APBC, Hong Kong, 14–17 Jan, 2007; Volume 5, pp. 27–36.
22. Jensen, K.L.; Styczynski, M.P.; Rigoutsos, I.; Stephanopoulos, G.N. A generic motif discovery algorithm for sequential data. *Bioinformatics* **2006**, *22*, 21–28.
23. Leslie, C.S.; Eskin, E.; Cohen, A.; Weston, J.; Noble, W.S. Mismatch string kernels for discriminative protein classification. *Bioinformatics* **2004**, *20*, 467–476.
24. Dipartimento Di Ingegneria Dell'Informazione. Available online: <http://www.dei.unipd.it/~ciompin/main/filtering.html> (accessed on 21 May 2013).
25. Apostolico, A.; Comin, M.; Parida, L. Conservative extraction of over-represented extensible motifs. *Bioinformatics* **2005**, *21*, 9–18.
26. Mendes, N.D.; Casimiro, A.C.; Santos, P.M.; Sá-Correia, I.; Oliveira, A.L.; Freitas, A.T. MUSA: A parameter free algorithm for the identification of biologically significant motifs. *Bioinformatics* **2006**, *22*, 2996–3002.
27. Peng, C.H.; Hsu, J.T.; Chung, Y.S.; Lin, Y.J.; Chow, W.Y.; Hsu, D.F.; Tang, C.Y. Identification of degenerate motifs using position restricted selection and hybrid ranking combination. *Nucleic Acids Res.* **2006**, *34*, 6379–6391.
28. Vishnevsky, O.V.; Kolchanov, N.A. ARGO: A web system for the detection of degenerate motifs and large-scale recognition of eukaryotic promoters. *Nucleic Acids Res.* **2005**, *33*, W417–W422.
29. Chakravarty, A.; Carlson, J.M.; Khetani, R.S.; DeZiel, C.E.; Gross, R.H. SPACER: Identification of *cis*-regulatory elements with non-contiguous critical residues. *Bioinformatics* **2007**, *23*, 1029–1031.
30. Wu, R.; Chaivorapol, C.; Zheng, J.; Li, H.; Liang, S. fREDUCE: Detection of degenerate regulatory elements using correlation with expression. *BMC Bioinforma.* **2007**, *8*, doi:10.1186/1471-2105-8-399.

31. Wang, G.; Yu, T.; Zhang, W. WordSpy: Identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic Acids Res.* **2005**, *33*, W412–W416.
32. Ukkonen, E. Maximal and minimal representations of gapped and non-gapped motifs of a string. *Theoret. Comput. Sci.* **2009**, *410*, 4341–4349.
33. Romer, K.; Kayombya, G.R.; Fraenkel, E. WebMOTIFS: Automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches. *Nucleic Acids Res.* **2007**, *35*, W217–W220.
34. Zhang, S.; Su, W.; Yang, J. ARCS-Motif: Discovering correlated motifs from unaligned biological sequences. *Bioinformatics* **2009**, *25*, 183–189.
35. Coatney, M.; Parthasarathy, S. MotifMiner: A General Toolkit for Efficiently Identifying Common Substructures in Molecules. In Proceedings of the 3rd IEEE BIBE, Maryland, MD, USA, 10–12 March 2003; pp. 336–340.
36. Wijaya, E.; Yiu, S.M.; Son, N.T.; Kanagasabai, R.; Sung, W.K. MotifVoter: A novel ensemble method for fine-grained integration of generic motif finders. *Bioinformatics* **2008**, *24*, 2288–2295.
37. Tompa, M.; Li, N.; Bailey, T.L.; Church, G.M.; Church, G.M.; Moor, B.D.; Eskin, E.; Favorov, A.V.; Frith, M.C.; Fu, Y.; Kent, W.J.; *et al.* Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.* **2005**, *23*, 137–144.
38. Edwards, R.J.; Davey, N.E.; Shields, D.C. CompariMotif: Quick and easy comparisons of sequence motifs. *Bioinformatics* **2008**, *24*, 1307–1309.
39. Jiang, H.; Zhao, Y.; Chen, W.; Zheng, W. Searching Maximal Degenerate Motifs Guided by a Compact Suffix Tree. In *Advances in Computational Biology*; Arabnia, H.R., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 680, pp. 19–26.
40. Edelman, G.M.; Gally, J.A. Degeneracy and complexity in biological systems. *Proc. Natl. Acad. Sci. USA* **2001**, *98*, 13763–13768.
41. Shinozaki, D.; Akutsu, T.; Maruyama, O. Finding optimal degenerate patterns in DNA sequences. *Bioinformatics* **2003**, *19*, 206–214.
42. Bailey, T.L.; Williams, N.; Misleh, C.; Li, W.W. MEME: Discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Res.* **2006**, *34*, 369–373.

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).