2013 24th International Workshop on Database and Expert Systems Applications

Reducing the space of degenerate patterns in protein remote homology detection

Matteo Comin
Department of Information Engineering
University of Padova
Padova 35131, Italy
Email: comin@dei.unipd.it

Davide Verzotto
Computational and Systems Biology
Genome Institute of Singapore,
Singapore 138672, Singapore
Email: verzottod@gis.a-star.edu.sg

***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 outcome often far exceeds the size of the original input, making the output hard to be managed and then interpreted by refined analysis requiring manual inspection. In this article we discuss a characterization of degenerate patterns with character classes, and introduce the concept of *pattern priority*, for comparing and ranking different patterns without gaps, together with the class of *underlying patterns*, which permits to filter 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 sequences with remote homologies. Results show that our approach drastically reduces the number of patterns in output from a tool for protein sequence analysis, while retaining the functional ones.**
Availability: http://www.dei.unipd.it/~ciompin/main/filtering.html

I. 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, since such patterns usually correspond to residues conserved by evolution due to some significant structural or functional role [2], [3]. 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. Pattern discovery techniques have been used for a wide range of applications, from data compression [4] to protein family detection [5], [6].

In this article 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 in protein sequences.

A. Motivation: Clustering Degenerate Patterns

Degenerate patterns without gaps represent highly conserved segments of more complex signatures in protein families sharing remote homologies. To cope with the combinatorial explosion of degenerate patterns shared by a set of protein sequences, *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], [3], [7], [8]. Despite that, the number of patterns in output still remains intractable in most cases. Moreover, all pattern discovery tools must ultimately rank the output, according to some measure of importance.

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 as the fundamental units of functional signatures, e.g. *short linear motifs* (SLiMs), in order to identify those patterns representing the same loci, thus reducing the number of candidate solutions to be tested. In particular, we will follow and adapt the idea of pattern *minimality* introduced in [9] 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 input.

Recently, a number of ensemble methods addressing some of these clustering issues have been proposed [10]–[13]. 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. WebMotifs [10], ARCS-Pattern [11], MotifMiner [12], and MotifVoter [13] 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, in Tompa's benchmark [14] and Escherichia coli datasets, the average sensitivity is only improved by $62\,\%$, but the average precision is reduced by $15\,\%$ [13].

B. Problem Formulation

More formally, the problem we address is the following: *Given a reference sequence s, that is the concatenation of multiple protein sequences, and a set of* patterns with character classes $\mathcal{M}$, *that is the outcome of one or more pattern finders, reduce the set $\mathcal{M}$ to*

1529-4188/13 $26.00 © 2013 IEEE
DOI 10.1109/DEXA.2013.36

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	$[a,c,d]$	$[b,e]$

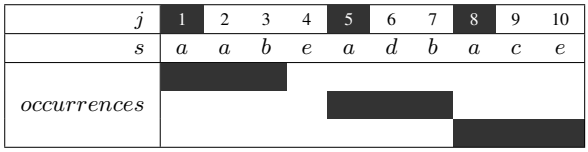


Fig. 1. Example of pattern. The occurrences of p in s are drawn in black.

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 article we discuss the basic problems behind degenerate pattern comparison and ranking in a simple and conservative fashion. In principle there are several ways to compare degenerate patterns, and they all use the number of occurrences and locations of patterns. Here we propose a simple yet effective way that will be compared with other traditional binary relations in Section V. There are mainly three features that characterize a degenerate pattern: length, degeneracy, and location list. A trade-off between these three features is required in order to compare and rank all patterns [15], [16]. We chose the combination, called *pattern priority*, that tries to represent most of the functional patterns present in PROSITE. An intuitive way to rank all patterns will be to favor degenerate patterns representing large matching regions between sequences, that are more likely to be functional [17], and thereafter those with the least degeneracy, which represents the degree of divergence during evolution of sequences and is "a prerequisite for and an inescapable product of the process of natural selection itself" [18].

II. 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 $s = s_1 s_2 \dots s_n$ 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 V we will clarify how to deal with multiple sequences. For the rest of the article, 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*. A character class C is a subset of Σ of cardinality at least two. Let's assume that we have several character classes C_j. We are interested to study sequences in the form of strings with character classes, where $\{2^{C_j}\}$ represents all non-empty subsets of the classes C_j.

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*. As of Figure 1, $s = aabeadbace$, $\Sigma = \{a, b, c, d, e\}$, $q = 3$, 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 $k = 3$. The main issue shown in 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].*

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

Fig. 2. Example of minimal representation of a pattern m with sequence p.

A sequence $p = p_1 p_2 \dots p_k$ 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 $1 \leq j \leq k$. 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 Figure 1 $m = (p, \mathcal{L}_m)$ 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 2 *(Pattern, Location list) We say that $m = (p, \mathcal{L}_m)$ is a pattern with sequence $p = p_1 p_2 \dots p_k$ and location list $\mathcal{L}_m = (l_1, l_2, \dots, l_\nu)$, IFF: (i) $|p| \geq 2$; (ii) $|\mathcal{L}_m| \geq q$; and (iii) $\mathcal{L}_m$ is complete.*

III. 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 3 *(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 $1 \leq j \leq k$.*

Proposition 1 *The minimal representation of a sequence is unique.*

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 the previous observation suggests us that $\mu(m)$ agrees with the definition of pattern (that is, $\mathcal{L}_{\mu(m)}$ is complete):

Definition 4 *(Minimal pattern) The minimal representation of m, given by $\mu(m) = (\mu(p), \mathcal{L}_m)$, is called a* minimal pattern*.*

To have a more concrete idea about this concept, Figure 2 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.

Let $\mathcal{M}$ be a set of patterns with character classes lying on the string s. From Fact 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.

We call $\mu(\mathcal{M})$ the *minimal set* of $\mathcal{M}$. Since mapping $\mathcal{M}$ in $\mu(\mathcal{M})$ can bring to a drastic reduction of 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. 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 chose a very intuitive combination of pattern length, pattern degeneracy, and location list.

Definition 5 *(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.*

Theorem 1 *Given any set of patterns $\mathcal{M}$, its minimal set $\mu(\mathcal{M})$ is totally ordered under the binary relation of pattern priority.*

It can be easily checked that the binary relation of *pattern priority* is irreflexive and antisymmetric. Furthermore Theorem 1 can be proved as a consequence of the results reported in [19], [20]. From Theorem 1 we have that 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.

IV. Underlying Patterns

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 lie 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 6 *(Underlying Pattern) The set of patterns $\mathcal{U} \subseteq \mu(\mathcal{M})$ is said to be* underlying *IFF:*

(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 $\mathcal{U}$.

Algortihm 1
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 a location vector Γ_s in the positions where m appears.*

The correctness of the algorithm follows from Theorem 1. 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_s, that represents the locations of s. This means that, $\forall\, l \in \mathcal{L}_m$, we store the value TRUE in the locations $\Gamma_s[l+i]$, 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_s 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 [20]. However this is out of scope for this article 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 $\mathcal{M}$.*

V. Experimental Results

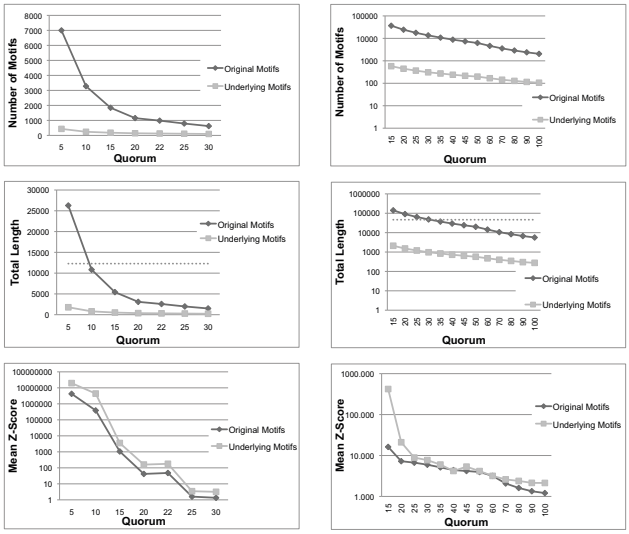
In this section we discuss the ability of underlying patterns to efficiently capture conserved degenerate patterns. More generally, there are two types of scenarios where the notion of underlying patterns could be useful. The first case is when a region of interest has already been identified, so that 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 [7], a tool for *de novo* pattern discovery. The dataset consists of six protein families for which Varun successfully extracts the functional patterns contained in the PROSITE signatures (release 20.85) [7]. 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*), composed by 22 sequences of 12,300 amino acids in total. This family contains two functional 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* (id PS01286; *Fa*), composed by 40 sequences of length 46,500 amino acids. 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; *Form*). *[LIVMA][LIVMY].G[GSTA][DES]L[FI][TN][GS]* is present in 17 sequences of length 5,300.
4) *Ubiquitin-activating enzyme* (id PS00865; *Ubi*). *P[LIVMG]CT[LIVM][KRHA].[FTNM]P* appears in 36 proteins of length 25,200.
5) *RNA polymerases M/15 Kd subunit* (id PS01030; *Poly*). *[FY]C.[DEKSTG]C[GNK][DNSA][LIVMHG][LIVM]* occurs in 29 sequences of length 4,000.
6) *Dbl homology (DH) domain* (id PS00741; *Dbl*). *[LM]..[LIVMFYWGS][LI]..[PEQ][LIVMRF]..[LIVM]*.

[KRS].[LT].[LIVM].[DEQN][LIVM]...[STM] appears in 65 sequences of length 18,750.

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 q refers to the number of occurrences in the concatenation of all sequences of a certain family, of total length n. Then, we employ the extracted patterns as input to our algorithm in order to compute the underlying patterns $\mathcal{U}$. All experiments were conducted on a common PC and the filtering process requires on average less than 10 seconds.



(a) Family Ni (b) Family Fa

Fig. 3. Total number, sum of lengths, and mean Z-Score of the patterns extracted using Varun and their corresponding underlying patterns, for Ni and Fa. The dashed line in 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.

For both sets of patterns $\mathcal{M}$ and $\mathcal{U}$ we compute some global statistics. Figure 3 shows the number of patterns, the sum of their lengths and the mean Z-Score (see [7]) for the first two protein families Ni and Fa. The other families share the same behavior (figure not shown). As expected, the number and sum of lengths of the underlying patterns is always much smaller than those of the original patterns, for which the sum of lengths abundantly exceeds the total length of sequences under examination in both families. Moreover, as seen above, the sum of lengths of the underlying patterns is always bounded by the length of sequences; 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. In Figure 3 we can see that, for all quorums, the average Z-Score of the 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 number and span of underlying patterns is much more manageable than the original set and also that their average quality, measured by the Z-Score, 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. We consider in detail again the first two families Fa and Ni and the corresponding functional signatures. The other four families show similar results and are summarized in Table III.

TABLE I
MAXIMUM SIMILARITY WITH THE REFERENCE PATTERNS FOR THE FAMILY Ni

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 II
MAXIMUM SIMILARITY WITH THE REFERENCE PATTERNS FOR THE FAMILY Fa

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

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 functional 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 I and II summarize the maximum similarity, for different quorums, of $\mathcal{M}$ and $\mathcal{U}$ with each functional signature of Ni and Fa, divided by an upper bound on the similarity with the original sets. For example, in the first row of Table I the quorum is 5. In this case, the maximum similarity of $\mathcal{M}$ with the functional 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. The values presented in Table I and II confirm that in most cases the functional patterns, that were present in $\mathcal{M}$, are also selected in the set of underlying patterns with a similar accuracy.

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 III reports the average scores for each measure for all six protein families, where a large maximum similarity with the two PROSITE functional signatures and a higher rank are preferable. In this context we consider different ranking methods: lexicographic order of patterns (*lexic.*), frequency ($|\mathcal{L}_m|$), inverse frequency, pattern probability assuming either an i.i.d. distribution of symbols based on amino acid frequencies, or an equal distribution of symbols (*probability no back.*), and Z-Score (computed as in [7]).

TABLE III
COMPARISON BETWEEN DIFFERENT BINARY RELATIONS APPLIED TO THE NOTION OF UNDERLYING PATTERNS

	$Ni_{1,2}$		$Fa_{1,2}$		*Form*	
Binary Relation	Sim.	Rank	Sim.	Rank	Rank	Sim.
Pattern priority	151/157	2.78	247/264	5.34	186/205	4.72
Z-Score	127/157	5.00	223/264	9.96	167/205	6.00
Probability	127/157	5.00	223/264	9.96	167/205	6.00
Probability no back.	127/157	5.00	223/264	9.96	165/205	6.20
Frequency	93/157	22.78	168/264	9.42	102/205	26.62
Inv. frequency	118/157	6.14	212/264	5.69	154/205	7.92
Lexic. order occ.	93/157	5.50	142/264	11.77	105/205	16.44

	Ubi		*Poly*		*Dbl*	
Binary Relation	Sim.	Rank	Sim.	Rank	Sim.	Rank
Pattern priority	190/198	3.40	215/234	4.25	498/522	5.20
Z-Score	178/198	4.74	212/234	5.86	455/522	7.10
Probability	178/198	4.74	210/234	5.92	455/522	7.10
Probability no back.	178/198	4.74	210/234	5.92	452/522	7.10
Frequency	112/198	9.75	135/234	13.69	321/522	21.35
Inv. frequency	159/198	6.00	188/234	10.10	436/522	13.74
Lexic. order occ.	112/198	12.39	126/234	13.93	308/522	22.00

We can easily see that the pattern priority applied to the notion of underlying patterns achieves the best scores among all methods for the detection of functional patterns. In addition, our heuristic ranks on average the reference patterns of Ni in the top 3 out of 2,239 candidate patterns in input, and those of Fa in the top 5 out of 10,842 patterns, while for the other families the reference patterns range from the top 3 to the top 5 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. [15].

VI. CONCLUSION

In this article we have studied patterns with character classes, introducing the notion of pattern priority and of underlying patterns. These notions has proved to be valuable for the analysis of biological sequences, bounding the total length of degenerate patterns in output from any modern pattern discovery tool. Preliminary experiments on protein families have shown a good performance of our approach as a filter to reduce the number of patterns in output, while keeping the functional ones.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, B. Cuche, E. de Castro, C. Lachaize, P. Langendijk-Genevaux, and C. Sigrist, "The 20 years of PROSITE," *Nucleic Acids Res.*, vol. 36, no. Database issue, pp. D245–D249, 2008.

[2] L. Parida, *Pattern Discovery in Bioinformatics: Theory and Algorithms*, ser. Mathematical and Computational Biology. Chapman and Hall/CRC, 2007.

[3] K. L. Jensen, M. P. Styczynski, I. Rigoutsos, and G. N. Stephanopoulos, "A generic motif discovery algorithm for sequential data," *Bioinformatics*, vol. 22, no. 1, pp. 21–28, 2006.

[4] A. Apostolico, M. Comin, and L. Parida, "Bridging lossy and lossless compression by motif pattern discovery," *Lecture Notes in Computer Science*, vol. 4123, pp. 793–813, 2006.

[5] M. Comin and D. Verzotto, "The Irredundant Class method for remote homology detection of protein sequences," *Journal of Computational Biology*, vol. 18, no. 12, pp. 1819–1829, dec 2011. [Online]. Available: http://dx.doi.org/10.1089/cmb.2010.0171

[6] ——, "Classification of protein sequences by means of irredundant patterns," *BMC Bioinformatics*, vol. 11, no. Suppl. 1, p. S16, 2010.

[7] A. Apostolico, M. Comin, and L. Parida, "VARUN: discovering extensible motifs under saturation constraints," *IEEE/ACM Trans. Comput. Biology Bioinformatics*, vol. 7, no. 4, pp. 752–762, Oct–Dec 2010.

[8] S. Sinha and M. Tompa, "Discovery of novel transcription factor binding sites by statistical overrepresentation," *Nucleic Acids Res.*, vol. 30, no. 24, pp. 5549–5560, 2002.

[9] E. Ukkonen, "Maximal and minimal representations of gapped and non-gapped motifs of a string," *Theoret. Comput. Sci.*, vol. 410, no. 43, pp. 4341–4349, 2009.

[10] 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 Res.*, vol. 35, no. Suppl. 2, pp. W217–W220, 2007.

[11] S. Zhang, W. Su, and J. Yang, "ARCS-Motif: discovering correlated motifs from unaligned biological sequences," *Bioinformatics*, vol. 25, pp. 183–189, Jan 2009.

[12] M. Coatney and S. Parthasarathy, "MotifMiner: a general toolkit for efficiently identifying common substructures in molecules," in *Proc. 3rd IEEE BIBE*. IEEE Comput. Soc., 2003, pp. 336–340.

[13] 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*, vol. 24, pp. 2288–2295, Oct 2008.

[14] M. Tompa, N. Li, T. L. Bailey, G. M. Church, and *et al.*, "Assessing computational tools for the discovery of transcription factor binding sites," *Nat. Biotechnol.*, vol. 23, no. 1, pp. 137–144, 2005.

[15] R. J. Edwards, N. E. Davey, and D. C. Shields, "CompariMotif: quick and easy comparisons of sequence motifs," *Bioinformatics*, vol. 24, no. 10, pp. 1307–1309, 2008.

[16] M. Comin and D. Verzotto, "Filtering degenerate patterns with application to protein sequence analysis," *Algorithms*, vol. 6, no. 2, pp. 352–370, 2013.

[17] H. Jiang, Y. Zhao, W. Chen, and W. Zheng, "Searching maximal degenerate motifs guided by a compact suffix tree," in *Advances in Computational Biology*, ser. Adv. Exp. Med. Biol., H. R. Arabnia, Ed. Springer, 2010, vol. 680, pp. 19–26.

[18] G. M. Edelman and J. A. Gally, "Degeneracy and complexity in biological systems," *PNAS*, vol. 98, no. 24, pp. 13 763–13 768, 2001.

[19] M. Comin and D. Verzotto, "Comparing, ranking and filtering motifs with character classes: application to biological sequences analysis," in *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, M. Elloumi and A. Y. Zomaya, Eds. Wiley, to appear, 2013, ch. 13.

[20] ——, "Alignment-free phylogeny of whole genomes using underlying subwords," *BMC Alg. Mol. Biol.*, vol. 7, p. 34, 2012.