VARUN: Discovering Extensible Motifs under Saturation Constraints

Alberto Apostolico, Matteo Comin, and Laxmi Parida

***Abstract*— The discovery of motifs in biosequences is frequently torn between the rigidity of the model on the one hand and the abundance of candidates on the other. In particular, motifs that include wildcards or "dont cares" escalate exponentially with their number, and this gets only worse if a dont care is allowed to stretch up to some prescribed maximum length. In this paper, a notion of *extensible* motif in a sequence is introduced and studied, which tightly combines the structure of the motif pattern, as described by its syntactic specification, with the statistical measure of its occurrence count. It is shown that a combination of appropriate saturation conditions and the monotonicity of probabilistic scores over regions of constant frequency afford us significant parsimony in the generation and testing of candidate overrepresented motifs.**

A suite of software programs called *Varun*[1] is described, implementing the discovery of extensible motifs of the type considered. The merits of the method are then documented by results obtained in a variety of experiments primarily targeting protein sequence families. Of equal importance seems the fact that the sets of all surprising motifs returned in each experiment are extracted faster and come in much more manageable sizes than would be obtained in the absence of saturation constraints.

***Index Terms*— Computational genomics, pattern discovery, data mining, motif, protein sequence, protein family.**

I. Introduction

THE discovery of motifs in bio-sequences is attracting increasing interest due to the perceived multiple implication of motifs in biological structure and function. The approaches to motif discovery may be partitioned in two main classes. In the first class, the sample string is tested for occurrences of motifs in a family of *a priori* defined, abstract *models* or templates. The second class of approaches assumes that the search may be limited to substrings in the sample or to some more or less controlled neighborhood of those substrings. The approaches in the first class are more rigorously justifiable, but often pose daunting computational burdens. Those in the second class tend to be computationally viable but rest on more shaky methodological grounds.

The characterizations offered for the notion of a motif could be partitioned roughly into statistical and syntactic. In a typical statistical characterization, a motif is a sequence of m positions such that at each position each character from (some subset of) the alphabet may occur with a given probability or weight. This is often described by a suitable matrix or *profile*, where columns correspond to positions and rows to alphabet characters (see, e.g., [11], [13]). The lineage of syntactic characterizations could be ascribed to the theory of error correcting codes: a motif is a pattern w of length m and an occurrence of it is any string at a distance of d, the distance being measured in terms of errors of a certain type. E.g., we can have only substitutions in the *Hamming* variant, substitutions and indels in the *Levensthein* variant, and so on (see, e.g., [12], [16]). Syntactic characterizations enable us to describe the model of a motif, or a realization of it, or both, as a string or simple regular expression over an extension of the input alphabet Σ, e.g., over $\Sigma \cup \{.\}$, where "." denotes the "don't care" character.

Irrespective of the particular model or representation chosen, the tenet of motif discovery equates over-representation of a motif with surprise and hence with interest. Thus, any motif discovery algorithm must ultimately weigh motifs against some threshold, based on a score that compares empirical and expected frequency, perhaps with some normalization. The departure of a pattern w from expectation is commonly measured by so-called z-scores (see, e.g., [14]), which have the form

$$z(w) = \frac{f(w) - E(w)}{N(w)}$$

where $f(w) > 0$ represents a frequency, $E(w) > 0$ an expectation and $N(w) > 0$ is the expected value of some function of w. For given z-score function, set of patterns $\mathcal{W}$, and real positive *threshold* T, patterns such that $z(w) > T$ or $z(w) < -T$ are respectively dubbed *over-* or *under-represented*, or simply *surprising*. The problem is that the number of patterns extracted in this way may escalate quite rapidly, a circumstance that seems to preclude precisely those massive analyses that have become conceivable with the increasing availability of whole genomes. Large-scale statistical tables may not only impose unbearable computational burden. They are also impractical to visualize and use, a circumstance that may defy the purpose of building them in the first place. A little reflection establishes how exponential build-up may take place. Assume that on the binary alphabet both $aabaab$ and $abbabb$ are asserted as reflections of candidate interesting motifs. We can give a concise description of one such motif by writing $a.ba.b$, with "." denoting the don't care, and then look for further occurrences of it. By this, however, we have immediately annexed also the spurious patterns $aababb$ and $abbaab$. A similar problem presents itself in the approaches that resort to profiles or weighed matrices mentioned earlier. In all of these cases, the risk is having to tell Horatio that there are more things in his philosophy than are dreamed of in heaven and earth[2]. Even setting aside computational aspects, tables that are too large at the outset risk to saturate the visual bandwidth of

A. Apostolico is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280 USA *and* the Department of Information Engineering, University of Padova, Italy. M. Comin is with the Department of Information Engineering, University of Padova, Italy. L. Parida is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA.

[1]Varun, along with experiments and input files used in this paper, is available for use at:
`www.research.ibm.com/computationalgenomics`.
A preliminary version of this paper has been presented at ISMB 2005 [3].

[2]"There are more things in heaven and earth, Horatio, Than art dreamt of in your philosophy"- W. Shakespeare, Hamlet, I, v [76].

the user. In this spirit, approaches that limit from the start the number of patterns to be considered may reap a more significant throughput, even in the comparison with exhaustive methods.

We regard the motif discovery process as distributed on two stages, where the first stage unearths motifs endowed with a certain set of properties and the second filters out the interesting ones. Since the redundancy builds up in the first stage, it is there that we have to look for possible ways of reducing the unnecessary throughput. Since over-representation is measured by a score, one would have to find ways to neglect candidate motifs that cannot possibly make it to the top list, and ideally spot such motifs before they are even computed. Counterintuitive as it might look, we show that such a possibility may be offered by certain attributes of "saturation" that combine in a unique way the syntactic structure and the list of occurrences or frequency for a motif. With solid words (i.e. a string of solid characters), for example, we know that in the worst case the number of distinct substrings in a string can be quadratic in the length of that string. Yet, if we partition the substrings into buckets by putting in the same bucket strings that have exactly the same set of occurrences, then we only need a number of buckets linear in the textstring [7]. Similar linear bounds were established for special classes of *rigid* motifs containing "don't cares" [4]. When combined with intervals of score monotonicity, properties of this kind support the global detection of unusual words of any length in overall linear space [2]. Some of these conservative scoring techniques were extended recently to rigid motifs with a prescribed maximum number of mismatches or don't care [5].

In this paper, we introduce and study a characterization of *extensible* motifs in the definition of which structural or syntactic properties and occurrence statistics are solidly intertwined. We show that a prudent combination of saturation conditions (expressed in terms of minimum number of don't cares compatible with a given list of occurrences) and monotonicity of scores afford us significant parsimony in the generation and testing of candidate over-represented motifs. More specifically, we isolate as candidate surprising motifs only the members of an *a priori* well identified set of "maximal" or "saturated" patterns. By this set being identifiable *a priori* we mean that the motifs in the set can be known before any score is computed. By neglecting the motifs other than those in our set we would not be overlooking any surprising motif. In fact, we maintain that any such motif: (i) is embedded in one of the saturated ones, and (ii) does not achieve a larger score than the latter (hence, computing its score and publishing it explicitly would take more time and space but not add information). The results of this paper apply to *extensible* patterns a philosophy previously applied to *rigid* motifs described (1) by solid words [2] and (2) by words of some specified fixed length affected by a specified maximum number of errors [5]. The transition from rigid to extensible motifs requires the orchestration of substantially novel concepts and tools, resulting in an algorithm for the extraction and weighing of extensible motifs, and a suite of software programs implementing the whole. The merits of the method are tested on families of protein sequences, as is documented in the last part of the paper. In all cases tested, the motif reported in PROSITE as most important in terms of functional/structural relevance emerges either at the top or among the top ten or so of the (short) output list. Experiments related to the sensitivity and selectivity of the method are also reported.

Many pieces of software exist in literature to carry out the task of motif or pattern discovery on string data. As already mentioned in part, the guiding principles underlying these discovery tools have been categorized in different ways: supervised vs unsupervised, aligned vs unaligned, enumerative vs non-enumerative, combinatorial vs statistical (or learning) methods. Yet another cataloguing is based on the motif architecture: *solid* patterns, patterns with fixed size gaps, patterns with variable sized gaps, patterns with specific gapped structures. See [15] for further elaboration along these lines and for citations.

The method presented here is unsupervised, unaligned, enumerative and (strictly speaking) a combinatorial approach during the discovery phase. The spirit of the approach is along the lines of the *a priori* scheme in [1] that was adopted in the software tool presented in [17]. The present algorithm overcomes one of the weaknesses present in the latter in terms of fixed gap sizes for the pattern. The transfer of the *a priori* scheme to an algorithm to tackle variable gap sizes in the architecture of a pattern is non-obvious and this is one of the primary contributions of this paper. Additional non-trivial expansions of the scheme reside in the introduction of statistical measures and scoring functions the monotonicity of which is exploited in order to reduce the search space.

This paper is organized as follows. In the next section, we stipulate some basic definitions and concepts and then proceed to derive expressions for the probabilities and expected number of occurrence of a motif under simple probabilistic models. We further derive monotonicity properties that hold for related z-scores under the fairly acceptable assumption that the probability of a motif occurrence is less than $1/2$. In Section 3 we discuss our algorithm, its implementation and usage. Section IV, and the annex entitled "Supplemental Material" contain results from rather extensive experiments on protein and, occasionally, DNA sequences and sequence families. A brief section of conclusions summarize our findings.

II. Motifs, Expectations and Scores

Let m be a sequence of sets of characters from an alphabet $\Sigma \cup \{.\}$, where '.' $\notin \Sigma$ denotes a don't-care (*dot*, for short) and the rest are *solid* characters. We use σ to denote a singleton character or a subset of Σ. For character (sets) e_1 and e_2, we write $e_1 \preceq e_2$ if and only if e_1 is a don't care or $e_1 \supseteq e_2$.

The input string s is a sequence of characters from the alphabet Σ. Given an input string s and a positive integer k, $k \leq |s|$, a sequence m is a *motif* (extensible or rigid) of s with $|m| > 1$ and location list $\mathcal{L}_m = (l_1, l_2, \ldots, l_p)$, if both $m[1]$ and $m[|m|]$ are solid and $\mathcal{L}_m$ is the list of all and only the occurrences of m in s. A motif m occurs at position l on s if $m[j] \preceq s[l+j-1]$ holds for $1 \leq j \leq |m|$.

Allowing for spacers in a motif is what makes it extensible. Such spacers are indicated by annotating the don't care characters. Specifically, an annotated "." character is written as $.^{\alpha}$ where α is a set of positive integers $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ or an interval $\alpha = [\alpha_l, \alpha_u]$, representing all integers between α_l and α_u including α_l and α_u. Whenever defined, d will denote the maximum number of consecutive don't cares allowed in a motif. In such cases, for clarity of notation, we use the *extensible wild card* denoted by the dash symbol "-" instead of the annotated dot character, $.^{[1,d]}$ in the motif. Note that '-' $\notin \Sigma$. Thus a motif of the form $a.^{[1,d]}b$ will be simply written as a-b. A motif m is *extensible* if it contains at least one annotated dot, otherwise m is *rigid*. Given an extensible

motif m, a rigid motif m' is a *realization* of m if each annotated dot $.^{\alpha}$ is replaced by $l \in \alpha$ dots. The collection of all such rigid realizations of m is denoted by $R(m)$. For a *rigid* motif the set of its realizations contains only one element, the motif itself. An extensible motif m occurs at position l in s if there exists a realization m' of m that occurs at l. Note that an extensible motif m could possibly occur a *multiple number of times* at a location on a sequence s. Throughout in the discussion we are interested mostly in the (unique) first left-most possible occurrence at each location.

Given a motif m let $m[j_1]$, $m[j_2]$, ... $m[j_l]$ be the l solid elements in the motif m. Then the sub-motifs of m are given as follows: for every j_i, j_t, the sub-motif $m[j_i \ldots j_t]$ is obtained by dropping all the elements before (to the left of) j_i and all elements after (to the right of) j_t in m. A motif m' is a *condensation* of m, if m' can be produced by replacing some dots of m with solid characters or by extending m with new solid characters, eventually preceded by dots. This implies that m is a *condensation* for any of its sub-motifs.

We are interested in motifs for which any condensation would disrupt the cardinality of the list of occurrences. Formally, let m_1, m_2, ..., m_h be the motifs in a string s. A motif m_i is *maximal in length* if there exists no m_l, $l \neq i$ with $|\mathcal{L}_{m_i}| = |\mathcal{L}_{m_l}|$ and m_i is a sub-motif of m_l. A motif m_i is *maximal in composition* if no dot character of m_i can be replaced by a solid character that appears in all the locations in $\mathcal{L}_m$. If the motif m_i contains non-singleton characters then we require each sets of characters to be the smallest set, among the ones defined a priori by the user, that describes all occurrences of the motif. A motif m_i is *maximal in extension* if no annotated dot character of m_i can be replaced by a fixed length substring (without annotated dot characters) that appears in all the locations in $\mathcal{L}_m$. A motif that is maximal in composition, in extension and in length will be said to be *maximal* or *saturated*. Equivalently a motif m is *saturated* or *maximal* if any condensation of m would change the number of its occurrences.

We begin our treatment by deriving some simple expressions for the probability p_m of an extensible motif m under stationary, *iid* assumptions. Let m be an extensible motif generated by a stationary, *iid* source which emits $\sigma \in \Sigma$ with probability p_σ. Consider the set $R(m)$ of all possible realizations of m. Each realization is a string over $\Sigma \cup \{.\}$. For a specific realization $\overline{m}$, its probability $p_{\overline{m}}$ is given by

$$p_{\overline{m}} = \prod_{\sigma \in \Sigma} (p_\sigma)^{j_\sigma} \ , \tag{1}$$

where j_σ is the number of times σ appears in $\bar{m}$. Thus, the dot has implicitly probability 1.

An extensible motif is *degenerate* if it can possibly have multiple occurrences at a site i on the input s.

Lemma 1: Let m be an extensible non-degenerate motif generated by a stationary, *iid* source which emits ($\sigma \in \Sigma$) with probability p_σ. Let j_σ be the number of times σ appears in m and let e be the number of annotated dots in m with annotations $\alpha_1, \alpha_2, \ldots \alpha_e$. Then

$$p_m = \prod_{\sigma \in \Sigma} (p_\sigma)^{j_\sigma} \prod_{i=1}^{e} |\alpha_i| \ . \tag{2}$$

Proof. Since the motif is non-degenerate, by the definition of realization of a motif,

$$p_m = \sum_{\overline{m} \in R(m)} (p_{\overline{m}}) \ .$$

Hence we need to compute $p_{\overline{m}}$ where $\overline{m}$ is a rigid motif. Assume $\overline{m}$ is a rigid motif with no dot characters. By the *iid* assumption, $p_{\overline{m}} = \prod_{\sigma \in \Sigma} (p_\sigma)^{j_\sigma}$. Next, consider $\overline{m}$ to be a rigid motif with possibly some dot characters. Again, clearly, $p_{\overline{m}} = \prod_{\sigma \in \Sigma} (p_\sigma)^{j_\sigma}$. In other words, only the solid characters contribute non-trivially to the computation of $p_{\overline{m}}$. Hence, if m is not rigid,

$$p_m = |R(m)| \prod_{\sigma \in \Sigma} (p_\sigma)^{j_\sigma} \ .$$

But $|R(m)| = \prod_{i=1}^{e} |\alpha_i|$, whence the claim. □

Corollary 1: If m is a non-degenerate extensible motif where each $m[i]$ is a set of (homologous) characters, then

$$p_m = \prod_{m[i] \neq `.', `-'} \left(\sum_{\sigma \in m[i]} p_\sigma \right) \prod_{i=1}^{e} |\alpha_i| \ . \tag{3}$$

Let M^s denote a set of motifs that has only the solid characters of at least s occurrences of m. For example, consider the motif a-b with realizations $a.b$, $a..b$ and $a...b$. Then $M^1 = \{a.b, a..b, a...b\}$ since m occurs once on each $m \in M^1$; $M^2 = \{a.bb, a..bb, a.b.b\}$ since m occurs twice on each $m \in M^2$; $M^3 = \{a.bbb\}$ since m occurs three times on $m \in M^3$. More precisely, $M^s = \{m_i \mid m_i$ is the intersection of at least s realizations of m, $m_i \notin M^{s'} \ \forall 1 \leq s' < s\}$.

Corollary 2: Let m be a degenerate (possibly with multiple occurrences at a site) extensible motif, given r as the maximum number of multiple motif occurrences starting at same site, and let $p_{m^k} = \sum_{m' \in M^k} p_{m'}$, then

$$p_m = \sum_{k=0}^{r-1} (-1)^k \left(p_{m^{k+1}}\right) \ . \tag{4}$$

This follows directly from the inclusion-exclusion principle.

Notice that for a degenerate motif, Expression (2) is the zeroth order approximation of Expression (4). The first order approximation is $p_m \approx p_{m^1} - p_{m^2}$ and the second order approximation is $p_m \approx p_{m^1} - p_{m^2} + p_{m^3}$ and so on. Using Bonferroni's inequalities, a kth order approximation of p_m is an over-estimate of p_m, if k is odd. This is helpful in practice where we don't need to know the correct value of r.

Next, we obtain the form of p_m for a non-degenerate motif when input m is assumed to be generated by a Markov chain. For the derivation below, we assume the Markov chain has order 1. For further discussion, we introduce the following definition.

Definition 1: (cell $\langle \sigma_1, \sigma_2, \ell \rangle$, $C(m)$) A substring $\hat{m}$ on m is a cell, if $\hat{m}$ begins and ends in solid characters with only non-solid intervening characters: σ_1, at the start and σ_2 at the end position and ℓ is the number of intervening un-annotated dot characters. If the intervening character is the extensible character, then ℓ takes the value of -1. For convenience, the cell is represented by the triplet $\langle \sigma_1, \sigma_2, \ell \rangle$. $C(m)$ is the collection of all such cells of m. For example, using the above notation, $C(ab..c.d\text{-}g) = \{\langle a, b, 0 \rangle, \langle b, c, 2 \rangle, \langle c, d, 1 \rangle, \langle d, g, \text{-}1 \rangle\}$.

Let $p^{(k)}_{\sigma_1, \sigma_2}$ denote the probability of moving from σ_1 to σ_2 in k steps. Let s be a stationary, irreducible, aperiodic Markov chain of order 1 with state space Σ ($|\Sigma| < \infty$). Further, π_σ is the equilibrium probability of $\sigma \in \Sigma$ and the ($|\Sigma| \times |\Sigma|$) transition probability matrix $P[i, j]$ is defined as $p^{(1)}_{\sigma_i, \sigma_j}$. For a rigid motif $\overline{m}$,

for each cell $\langle \sigma_1, \sigma_2, \ell \rangle \in C(\overline{m})$ is such that $\ell \geq 0$. It is easy to see that when $\ell \geq 0$, the cell represents the (ℓ+1)-step transition probability given by $P^{\ell+1}$, i.e., $p_{\sigma_1(.)\ell\sigma_2} = P^{\ell}[\sigma_1, \sigma_2]$. Thus for a rigid motif $\overline{m}$ we have that

$$p_{\overline{m}} = \pi_{\overline{m}[1]} \prod_{\langle \sigma_1, \sigma_2, \ell \rangle \in C(\overline{m})} P^{\ell}[\sigma_1, \sigma_2] \ .$$

If m is an extensible motif on s then

$$p_m = \sum_{\overline{m} \in R(m)} \pi_{m[1]} \prod_{\langle \sigma_1, \sigma_2, \ell \rangle \in C(\overline{m})} P^{\ell}[\sigma_1, \sigma_2] \ . \tag{5}$$

In fact, using the definition of a realization yields

$$p_m = \sum_{\overline{m} \in R(m)} (p_{\overline{m}}) \ .$$

Hence we need to compute $p_{\overline{m}}$ where $\overline{m}$ is a rigid motif.

When sets of characters or homologous sets are used in motifs, the *cell* is appropriately defined so that σ_1 and σ_2 are sets of homologous characters, possibly singletons. Then the following holds.

Lemma 2: If m is an extensible motif where each $m[i]$ is a set of characters (homologous characters), then

$$p_m = \sum_{\overline{m} \in R(m)} \sum_{\overline{\sigma} \in \overline{m}[1] \times \ldots \times \overline{m}[|\overline{m}|]} \pi_{\overline{\sigma}[1]} \prod_{i=1}^{|\overline{m}|-1} \left(P^{\ell_i}[\overline{\sigma}[i], \overline{\sigma}[i+1]] \right) \tag{6}$$

where $\langle \overline{\sigma}[i], \overline{\sigma}[i+1], \ell_i \rangle$ is a cell of the realization $\overline{m}$.

When two motifs m_1 and m_2 both get a significant score (as defined in Expression (8)), and are very close to each other, it becomes desirable to discriminate one from the other. This calls for a more careful evaluation of the score.

In the previous definitions, we used the assumption that a motif is generated by a single stationary source. This model undergoes the restriction that also a mismatch is produced by such source, whereas in reality it is the concatenation of a series of events that generate this mismatch. We can revisit our earlier model and refine the treatment of the wild card under the i.i.d. assumption. In Lemma 1, the dot character is treated as "any" character emitted by the source and thus its probability is assigned to be 1. However, in computing the probability of the leftmost occurrence of a motif the dot character actually corresponds to a mismatch. A mismatch occurs when in comparing two input sequences at particular positions the two characters differ. We can express this probability as the complement of having two independent extractions from an urn return the same character, hence:

$$p_{\text{dot}} = 1 - \sum_{\sigma \in \Sigma} p_{\sigma}^2 \ ,$$

where $\sigma \in \Sigma$. Expression 2 can now be replaced by :

$$p_m = \prod_{\sigma \in \Sigma} (p_{\sigma})^{j_{\sigma}} \prod_{i=1}^{e} (|\alpha_i| p_{\text{dot}}^{\alpha_i}) \ . \tag{7}$$

Using $p_{\text{dot}} < 1$ instead of $p_{\text{dot}} = 1$ could be interpreted as a probabilistic way to include a "gap penalty" in the previous formulation. This new definition does not affect corollary [2].

Let now u and v be two motifs such that v can be obtained from u by a sequence of consecutive unit expansions –consisting each of adding a dot or a character or a character set, or replacing a dot character with a solid character or character set– that transforms u into v. In particular, any condensation v of u can be produced in this way. A score z is *monotonic* for u and v if the value of z is always either increasing or decreasing over any such expansion. The key observation here is that, under most probabilistic settings, the probability of a condensation v of u obeys $p_v \leq p_u$. This is almost immediate under i.i.d. distribution, as the following claim shows.

Theorem 1: Let v and u be possibly degenerate extensible motifs under the iid model and let v be a condensation of u. Then, there is a real $\hat{p} \leq 1$ such that $p_v = p_u \hat{p}$.

Proof. It is enough to consider the case of a unit condensation, i.e., where v has one more solid character than u. The claim holds trivially when the extra character is added at the beginning or at the end of u. In fact, in any such case the probability of the extra character multiplies each term of Expression 4, whence the whole probability as well. Consider next the case where the solid character in v substitutes a don't care of u. We begin by describing an alternate way to compute p_u. With ℓ denoting the length of a longest string in $R(u)$, compute the set of all strings over Σ^{ℓ} and store them consecutively row-wise in a table. Compute, for each row, the probability of the string in that row, which is the product of the probabilities of the individual characters (the sum of all row probabilities is 1). Consider now the realizations in $R(u)$ in succession. Check each realization against every row of the table; wherever the two match, mark the row if it had not been already marked. Let **R** be the set of rows that are marked at the outset. Clearly, adding up the probabilities of the rows in **R** yields p_u. Consider now the set of rows that would be similarly involved in the computation of p_v. This must be a subset of **R**, whence $p_v \leq p_u$. □

With Markov processes, the intuition at the basis is that if we split the transition probability into two consecutive segments then we have: $P^{\ell}[\sigma_1, \sigma_2] = \sum_{\sigma_k \in \Sigma} P^{\ell_1}[\sigma_1, \sigma_k] \times P^{\ell_2}[\sigma_k, \sigma_2]$, where $\ell = \ell_1 + \ell_2$. Since all $P^{\ell}[\sigma_i, \sigma_j] \geq 0$, then any specific character (or alphabet subset) acting as a bottleneck yields $P^{\ell}[\sigma_1, \sigma_2] \geq P^{\ell_1}[\sigma_1, \sigma_k] \times P^{\ell_2}[\sigma_k, \sigma_2]$. The following general property is derived in analogy with a similar one in [2].

Theorem 2: If $f(u) = f(v) > 0$, $N(v) < N(u)$, and $E(v)/N(v) \leq E(u)/N(u)$, then

$$\frac{f(v) - E(v)}{N(v)} > \frac{f(u) - E(u)}{N(u)} \ .$$

Proof. Multiplying both terms by $N(v)/E(v)$ and using the assumption $f(v) = f(u) \geq 0$ we get, after rearrangement

$$\frac{f(u)}{E(v)} \left(1 - \frac{N(v)}{N(u)} \right) > 1 - \frac{E(u)N(v)}{E(v)N(u)} \ .$$

Since $0 < N(v)/N(u) < 1$, then the left hand side is always positive. The right hand size is always negative or zero. □

When $N(u)$ is the square root of the variance, the z-score takes up the form

$$z(u) = \frac{f(u) - E(u)}{\sqrt{Var(u)}} \ .$$

In the Bernoulli model, for instance, this variance results in $\sqrt{np_u(1-p_u)}$. In our case, we let p_m be the probability of the motif m occurring at any location i on the input string s with $n = |s|$ and let k_m be the observed number of times it occurs on s. When it can be assumed that the occurrence of a motif m at a site is an *iid* process, (cf. [19], Chapter 12), we have for large n

and $k_m \ll n$,

$$\frac{k_m - np_m}{\sqrt{np_m(1-p_m)}} \to \mathcal{N}(0,1) \ . \tag{8}$$

Theorem 3: Let u and v be motifs generated with respective probabilities p_u and $p_v = p_u\hat{p}$ according to an iid process. If $f(u) = f(v)$ and $p_u < 1/2$ then

$$\frac{f(v) - E(v)}{\sqrt{E(v)(1-p_v)}} \geq \frac{f(u) - E(u)}{\sqrt{E(u)(1-p_u)}} \ .$$

Proof. We show that the functions $N(u) = \sqrt{E(u)(1-p_u)}$ and $E(u)/N(u)$ satisfy the conditions of Theorem 2. First, we prove that $E(v) \leq E(u)$. Indeed, since $(|v| - |u|)/(n - |u| + 1) \geq 0$,

$$\frac{E(v)}{E(u)} = \frac{(n-|v|+1)p_v}{(n-|u|+1)p_u} = \left(1 - \frac{|v|-|u|}{n-|u|+1}\right)\hat{p} \leq \hat{p} \leq 1 \ .$$

Next, we study the ratio

$$\left(\frac{N(v)}{N(u)}\right)^2 = \left(1 - \frac{|v|-|u|}{n-|u|+1}\right)\frac{p_v(1-p_v)}{p_u(1-p_u)} \leq \frac{p_v(1-p_v)}{p_u(1-p_u)} \ .$$

The concave product $p_u(1-p_u)$ reaches its maximum for $p_u = 1/2$. Since we assume $p_u < 1/2$, the rightmost term is smaller than 1. The monotonicity of $N(u)$ is satisfied.

Finally, we prove that also $E(u)/N(u)$ is monotonic, i.e., that $E(v)/N(v) \leq E(u)/N(u)$, which is equivalent to

$$\frac{E(v)}{E(u)} \ \frac{1-p_u}{1-p_v} \leq 1 \ .$$

But $E(v)/E(u) \leq 1$ by hypothesis and $(1-p_u)/(1-p_v) < 1$ since $p_u > p_v$. □

In conclusion, we can restrict our z-score computation to classes of maximal motifs, i.e., only compute the z-score for the maximally saturated motif among those in each class of motifs sharing the same list of occurrences.

The previous z-score is not the only way to measure events that occur with unexpected frequency. In applications related to classification and clustering, such as, e.g., with protein families, a motif m is considered to be over-represented if a surprisingly large number of sequences from an ensemble contain each *at least* one occurrence of m. In this context, a large total number of occurrences of m in any particular sequence is immaterial and may be misleading as a measure, since the relevant fact is that the motif is shared across multiple sequences.

Let p_m be the probability assigned to motif m, computed according to any of the models above. Assuming t sequences $s_1, s_2, ..., s_t$ to be given, the expected number of occurrences of the motif in s_i is approximately $\mu_i = p_m|s_i|$. By the law of rare events (Poisson distribution), the probability of finding m at least once in s_i is $p^{(i)} = 1 - e^{-\mu_i}$. The expected number of sequences containing m at least once is then

$$E = \sum_{i=1}^{t} p^{(i)} = t - \sum_{i=1}^{t} e^{-\mu_i} \ .$$

We can assess the statistical significance of a given discrepancy between observed and estimated by taking the χ-square ratio:

$$\frac{(Q-E)^2}{E}$$

where Q corresponds to the counted *quorum*, i.e., the number of sequences that contain m.

This setting does not alter the considerations on saturation and monotonicity developed earlier: the intuition is that, quorum being equal, a saturated motif will have a smaller probability and therefore its degree of over-representation would only increase.

III. Algorithmic Implementation

The algorithm implementing the above criteria works by iterated pairwise combination of segments of maximal extensible motifs, followed by pruning of those pairings that are found to be not maximal. The input is a string s of size n and two positive integers, K and D. The extensibility parameter D is interpreted in the sense that up to D (or 1 to D) number of dot characters between two consecutive solid characters are allowed. The output is all maximal extensible (with D spacers) motifs that occur at least K times in s. Incidentally, the algorithm can be adapted to extract rigid motifs as a special case. For this, it suffices to interpret D as the maximum number of dot characters between two consecutive solid characters.

The algorithm works by converting the input into a sequence of possibly overlapping *cells* (see Definition 1). A maximal extensible motif is a sequence of cells.

Initialization Phase

The cell is the smallest *extensible* component of a maximal pattern and the motif can be viewed as a sequence of overlapping cells. The initialization phase has the following steps.

Step 1: Construct patterns that have exactly two solid characters in them and separated by no more than D spaces or "." characters. This is done by scanning the string s from left to right. Further, for each location we store start and end position of the pattern. For example, if $s = abzdabyxd$ and $K = 2, D = 2$, then all the patterns generated at this step are: ab, $a.z$, $a..d$, bz, $b.d$, $b..a$, zd, $z.a$, $z..b$, da, $d.b$, $d..y$, $a.y$, $a..x$, by, $b.x$, $b..d$, yx, $y.d$, xd, each with its occurrence list. Thus $\mathcal{L}_{ab} = \{(1,2),(5,6)\}$, $\mathcal{L}_{a.z} = \{(1,3)\}$ and so on.

Step 2: The extensible cells are constructed by combining all the cells with at least one dot character and the same start and end solid characters. The location list is updated to reflect the start and end position of each occurrence. Continuing the previous example, b–d is generated at this step with $\mathcal{L}_{b-d} = \{(2,4),(6,9)\}$. All cells m with $|\mathcal{L}_m| < K$ are discarded. In the example, the only surviving cells are ab, b–d with $\mathcal{L}_{ab} = \{(1,2),(5,6)\}$ and $\mathcal{L}_{b-d} = \{(2,4),(6,9)\}$

Iteration Phase

Let B be the collection of cells. $Extract(B)$ is a routine that returns the first element of B in alphabetic order. If $m = Extract(B)$, then $m \in B$ and there does not exist $m' \in B$ such that $m' \succ m$ holds: $m_1 \succ m_2$ if one of the following holds: (1) m_1 has only solid characters and m_2 has at least one non-solid character; (2) m_2 has the "–" character and m_1 does not; (3) m_1 and m_2 have $d_1, d_2 > 0$ dot characters respectively and $d_1 < d_2$.

Further, m_1 is $\sim$-compatible with m_2 if the last solid character of m_1 is the same as the first solid character of m_2. Moreover if m_1 is $\sim$-compatible with m_2, then $m = m_1 \sim m_2$ is the concatenation of m_1 and m_2 with an overlap character, respectively at the common end and start character and $\mathcal{L}'_m = \{(x,y)|(x,l) \in \mathcal{L}'_{m_1}, (l,y) \in \mathcal{L}'_{m_2}\}$. For example if $m_1 = ab$ and $m_2 = b.d$ then m_1 is $\sim$-compatible with m_2 and $m_1 \sim m_2 = ab.d$. However, m_2 is not $\sim$-compatible with m_1.

Note that it is possible to have simultaneously m_1 $\sim$-compatible with m_2 and m_2 $\sim$-compatible with m_1 hold simul-

taneously and hence both conditions are checked in tandem. In particular we first try to extend m_1 to the right till the maximality condition is preserved and then reiterate the process to the left. For consistency the order of these two extensions must be always the same.

The procedure is best described by the pseudocode shown here. $NonMaximal(m)$ is a routine that checks if the new motif m is non-maximal w.r.t. motifs in $Result$ by checking the location lists; more precisely, the motif m is non-maximal if its location list can be constructed by merging and/or shifting other location lists of previously computed motifs. Steps G:18 and G:19 detect the suffix motifs of already detected maximal motifs. $Result$ is the collection of all the maximal extensible motifs.

Main()
{
  $Result \leftarrow \{\}$;
  $B \leftarrow \{m_i | m_i$ is a cell with at most D spaces$\}$;
  For each $m = Extract(B)$
    Iterate$(m, B, Result)$;
}
Iterate$(m, B, Result)$
{
G:1 $m' \leftarrow m$;
G:2 For each $b = Extract(B)$ with
G:3 (($b \sim$–*compatible* m') OR ($m' \sim$–*compatible* b))
G:4 If ($m' \sim$–*compatible* b)
G:5 $m_t \leftarrow m' \sim b$; compute $\mathcal{L}_{m_t}$;
G:6 If $NonMaximal(m_t)$ continue;
G:7 If ($|\mathcal{L}_{m'}| = |\mathcal{L}_b|$) $B \leftarrow B - \{b\}$;
G:8 If ($|\mathcal{L}_{m'}| \geq K$)
G:9 $m' \leftarrow m_t$;
G:10 **Iterate**$(m', B, Result)$;
G:11 If ($b \sim$–*compatible* m')
G:12 $m_t \leftarrow b \sim m'$; compute $\mathcal{L}_{m_t}$;
G:13 If $NonMaximal(m_t)$ continue;
G:14 If ($|\mathcal{L}_{m'}| = |\mathcal{L}_b|$) $B \leftarrow B - \{b\}$;
G:15 If ($|\mathcal{L}_{m'}| \geq K$)
G:16 $m' \leftarrow m_t$;
G:17 **Iterate**$(m', B, Result)$;
G:18 For each $r \in Result$ with $\mathcal{L}_r = \mathcal{L}_{m'}$
G:19 If (m' is not maximal w.r.t. r) continue;
G:20 $Result \leftarrow Result \cup \{m'\}$;
}

Correctness follows from the observation that the above procedure essentially constructs the inexact suffix tree of [10] implicitly, in a different order. A tight time complexity is more difficult to come by, however, if we consider M to be the number of extensible maximal motifs and S to be the size of the output – i.e. the sum of the sizes of the motifs and the sizes of the corresponding location lists – then the time taken by the algorithm is $O(SM \log M)$. The intuition behind this upper-bound is that maximal motifs are constructed incrementally by merging incomplete maximal motifs. By construction, we do not need to backtrack and the merging step can be performed efficiently using a hash table that maintains the motifs in sorted ordered. This results a log-rump construction that accounts for the $M \log M$ term. In experiments of the kind described later in the paper, at 3 GHz clock, time ranged typically from few minutes to half an hour.

A. Varun Implementation and Usage

In this section we give details related to the usage of *Varun*[3], the software suite implementing the discovery process described in the previous section, expanded by suitable combinatorial and statistical pruning.

Since the pattern space can vary dramatically for different classes of inputs, a number of parameters have been introduced to allow the user to best exploit his specific domain knowledge. By fine tuning some parameters, the user can variously prune the pattern space. There are essentially two classes of pruning parameters: (1) combinatorial and (2) statistical. To avoid clutter, we describe only a few of the critical pruning parameters here. Each parameter has a default value and it is not mandatory to specify them all.

Combinatorial pruning

1) Pruning by Occurrences:
   a) `-k<Num>`: `Num` is the quorum or the minimum number of times a pattern must occur in the input.
   b) `-c`: When this is specified the quorum `k` is in terms of the number of sequences where the pattern occurs at least once. For example if this option is set and further `-k10` is specified, then a valid pattern must occur in at least 10 distinct sequences. However if this option is not set then a valid pattern must have at least 10 occurrences, not necessarily in distinct sequences.
2) Pruning by composition:
   a) Using groups of equivalent symbols:
      i) `-b<File>`: `File` list of groups of symbols that can be considered equivalent. The default file is an empty file.
      ii) `n<Num>`: `Num` is the maximum number of bracketed elements (equivalence classes) in a pattern. For example, if "-n2" is specified, then $[IL]...[LV]$, $L.[LV]$-V are valid patterns but not $[LV][IL][LV]..L$.
   b) `-R`: When this mode is specified, only rigid patterns are discovered.
   c) Extensibility: The following two parameters are used to prune the space of extensible patterns. Table I shows an example of the size of the pattern space for different parameter values.
      i) `-D<Num>`: `Num` is the maximum number of consecutive don't care characters ('.') in the realization of an extensible pattern. Note that a don't care character and an extensible character are never consecutive in any valid pattern. For example, if "-D3" is specified, then $L...V$, LV, $L.L.V$ are valid patterns but not $L....L$. Further, an extensible pattern of the form L-V implies that there are one to three don't care characters in the occurrences of this pattern between the amino acids L and V.
      ii) `-d<Num>`: `Num` is the minimum number of non-extensible characters (including the don't care character) between two consecutive extensible characters ('-'). For example, if "-d4" is specified, then $L..H$-$L..H$-L is a valid pattern but not $L...H$-$L.H$-L.

[3]A character from Indian mythology who is thousand eyed and sees all that happens in the world.

TABLE I
NUMBERS OF PATTERNS IN THE EXPERIMENT IN FIGURE 5 WITH Z-SCORE ≥ 100.0 AT VARIOUS VALUES OF PARAMETERS D AND d WITH QUORUM $k = 53$

		D			
		2	3	4	5
d	3	121	196	370	1145
	4	121	194	355	1008
	5	114	182	326	891
	8	112	178	313	758
	10	112	178	313	727

Pattern	Probability	Occ.	Z-Score
[LIVP]-[LM]R.[GE][LIVP].GC	2,05647e-07	57	585,494
LR.[GE][LIVP].GC	2,53136e-07	63	582,758
L..[GE][LIVP].GC	4,77614e-06	70	148,626
R-[GE][LIVP].GC	6,33367e-06	66	121,48
L-[GE][LIVP].GC	1,43284e-05	83	101,21
G[LIVP][GE].GC	3,98344e-05	77	55,359
R-[LIVP].GC	4,68467e-05	65	42,6968
L-[LIVP].GC	0,00010598	112	48,3873

Fig. 1. A statistical summary of a small set of valid patterns on the Coagulation factors 5/8 type C domain, also used in Figure 5.

Statistical pruning

1) `-p<File>`: `File` lists the symbol probabilities used for the probabilistic analysis.
2) `-z<Val>`: `Val` is the minimum absolute value of Z-score of the patterns.

Information display

1) Displaying occurrence information: The different modes of displaying the occurrence list of each valid pattern are as follows. (1) The occurrence list is not displayed (option `-L0`). (2) Only the start position of each occurrence is displayed (option `-L1`). (3) The start and end position of each occurrence is displayed as $x_1 - x_2$ where x_1 is the starting position and x_2 the end position(option `-L4`).
2) Displaying statistical information: The different statistical information displayed for possible use are (see Section II) (1) the probability of occurrence of a pattern, (2) the observed number of occurrences and (3) the Z-score. Figure 1 shows an example.

IV. TESTS AND EXPERIMENTS

We tested Varun on several protein families by seeking the *surprising* motifs in each. Each family was picked at random from the PROSITE database. Here we report four cases, more families are included in the supplemental material.

1) Streptomyces subtilisin-type inhibitors (*id PS00999*). Bacteria of the Streptomyces family produce a family of proteinase inhibitors characterized by their strong activity toward subtilisin. They are collectively known as SSI's: Streptomyces Subtilisin Inhibitors. Varun discovers this functionally significant motif as the top ranking one out of 470 extensible motifs that satisfy the k, D, and d requirements (Figure 2).
2) Nickel-dependent hydrogenases (*id PS00508*). These are enzymes that catalyze the reversible activation of hydrogen and is further involved in the binding of nickel. Again, this functionally significant motif is detected in the top three by Varun out of 4150 extensible motifs (Figure 3).
3) G-protein coupled receptors family 3 (*id PS00980*). Varun finds that the most important structural motif in this family is in the top thirty of the motifs out of 3508 extensible motifs (Figure 4).
4) Coagulation factors 5/8 type C domain (FA58C) (*id PS01286)*. Varun places the most important structural and functional motif in this family is in the top two of the motifs out of 80290 extensible motifs (Figure 5).

Rank	z-score	Motif
1	**7,60E+07**	**RA.T[LV].C.P-(2,3)G.HP....AC[ATD].L....[ASG]**
2	21416,8	A..[LV].C.P-(2,3)G.HP-(1,2,4)[ASG].[ATD]
3	8105,33	A-(1,4)T....P-(2,3)G.HP....[ATD]-(3)L....[ASG]
4	5841,85	[ATD].T....P-(1,2,3)G.HP-(1,2,4)A.[ATD]
5	4707,62	P.[ASG]-(2,3,4)P....AC[ATD].L....[ASG]
6	4409,21	A..[LV]...P-(2,3)G.HP-(1,2,4)A.[ATD]
7	3086,17	P-(1,2,3)[ASG]..P-(4)AC[ATD].L....[ASG]
8	3068,18	R..[ATD]....P-(2,3)G.HP-(1,2,4)[ASG].[ATD]
9	2615,98	[ASG][ATD]-(1,3,4)P....AC[ATD].L....[ASG]
10	2569,66	[ASG]-(1,2,3,4)P....AC[ATD].L....[ASG]
11	2145,6	G-(2,3)P....AC[ATD].L....[ASG]

Fig. 2. The functionally relevant motif is shown in bold for Streptomyces subtilisin-type inhibitors signature (id PS00999). Here 20 sequences of about 2500 amino acids were analyzed at k=20, D=4, d=4.

Rank	z-score	Motif
1	**295840**	**[LIM]-(1,2,3,4)[STA][FY]DPC[LIM][ASG]C[ASG].H**
2	**286535**	**[LIM]-(1,2,3,4)[ASG][FY]DPC[LIM][ASG]C[ASG].H**
3	**155736**	**R-(1,4)[FY]DPC[LIM][ASG]C[ASG].H**
4	78829	[LIM]-(1,2,3,4)[STA].DPC[LIM][ASG]C[ASG].H
5	76101,9	[LIM]-(1,2,3,4)[ASG].DPC[LIM][ASG]C[ASG].H
6	34205,6	[STA]-(1,4)DPC[LIM][ASG]C[ASG].H
7	30325,1	[LIM]-(1,2,3,4)[STA][FY]D.C[LIM][ASG]C..H
8	29276	[LIM]-(1,2,3,4)[ASG][FY]D.C[LIM][ASG]C..H
9	20527,3	[ASG]-(1,4)DPC[LIM][ASG]C[ASG].H
10	17503,4	[LIM]-(1,2,3,4)[ASG]..PC[LIM][ASG]C[ASG].H

Fig. 3. The functionally relevant motifs are shown in bold for Nickel-dependent hydrogenases (id PS00508). Here 22 sequences of about 23000 amino acids were analyzed at k=22, D=4, d=3.

To summarize, we find that in almost all cases, the motif documented as the most important (as functionally/structurally relevant) motif in PROSITE is in the top extensible motifs returned by Varun as *surprising*. In the third set (Figure 4) we find the PROSITE motif at position 42. This experiment shows that in some particular cases the patterns reported by Varun can be grouped together, in fact the top scoring motifs are very close to each other in location and composition. This suggests that a post processing step that clusters together the top patterns could improve the goodness of the results. In all cases, the difference in the z-score between the top few and the rest is dramatic as can be seen in Figures 2 through 5. The differing values of the Z-scores of each family is attributed to the different sizes of the families (the number of members and the length of each member). When testing with unrelated sequences the distribution of z-scores shows much smaller peaks compared to a family of proteins. To test if the values of Z-scores presented in the figures are significant w.r.t. to a set of random sequences, for each experiment we generate a set of random sequences with exactly the same number of sequences and the same lengths. Then we extract motifs using the settings of the various experiments to further analyze the Z-scores distribution. In all cases the Z-score ranges between 0 and 38 with

Rank	z-score	Motif
1	2,84E+09	Y...L...C..[FYW]A..[STAH]R..P..FNE[STAH]K.I.F[STAH]M
2	8,28E+07	V-(1,3,4)G...S..[STAH]....N...L....Q-(4)[STAH]....L.[DN]...[FYW]..F....P....Q..A...I
3	5,55E+07	L-(2,3)F...Q....[STAH][STAH]...L..[DN]...[FYW]..F.R..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	F.I-(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	F.I...[STAH]-(3)[STAH]....L.[DN]...[FYW]..F....P.D..Q..A...I
40	1,97E+06	F.I-(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

Fig. 4. The functionally relevant motif is shown in bold for G-protein coupled receptors family 3 (id PS00980). This run involved 25 sequences of about 25000 amino acids each at k=25, D=4, d=8.

Rank	z-score	Motif
1	**969,563**	**P-(4,5,8,9,10)[LM]R.[GE][LIVP].GC**
2	694,1	P-(4,5,8,9,10)[LM]R.[GE][LIVP].[GE]C
3	370,594	[LIVP]-(1,3,4,5,6,7,8,9,10)[LM]R.[GE]..[GE]C
4	361,052	P-(4,5,8,9,10)[LM]R.[GE]..[GE]C
5	261,519	[LIVP]-(1,3,4,5,6,7,8,9,10)[LM]R.[GE][LIVP]..C
6	261,519	[LIVP]-(1,3,4,5,6,7,8,9,10)[LM]R..[LIVP].[GE]C
7	254,971	P-(4,5,8,9,10)[LM]R.[GE][LIVP]..C
8	254,971	P-(4,5,8,9,10)[LM]R..[LIVP].[GE]C
9	249,763	[LIVP].....[LIVP]-(1,2,4,5,6,7,8,9,10)R.[GE]..GC

Fig. 5. The functionally relevant motif is shown in bold for Coagulation factors 5/8 type C domain (id PS01286). Here 40 sequences of about 80290 amino acids were analyzed. Notice that in this case, the motifs have a fairly large gap size of 10 amino acids at k=40, D=10, d=10.

a strong bias toward values smaller than 10. Clearly no signal was present in the data, thus motifs with a Z-score in this range can be discarded. Similar experiments can be conducted to find the best setting for each set of sequences. In general we choose k to be the total number of sequences, since we are looking for a signal that is shared by all sequences. The don't care parameters D and d are obtained after experimenting with the data. In most cases different combinations of D and d can discover the functional motif or a more specific version of it. A similar behavior is also observed with rigid motifs (see next).

Next, we test the sensitivity and selectivity of Varun using the families as reported in PROSITE. Since most of the family sizes are small, we do these experiments along the lines in [18] page 46. The following six sets were selected, randomly in each family: 5 sequences in each of the families, High potential iron-sulfur proteins, Streptomyces subtilisin-type inhibitors, Nickel-dependent hydrogenases, G-protein coupled receptors family 3 and Coagulation factors 5/8 type C domain, and 8 sequences from the family of Chitin-binding type-1 domain. First, each family was contaminated with one of the sets that was drawn from a different family (for example the five sequences of G-protein was mixed with the family of the hydrogenases). Next we contaminated each family with two sets from a different family and then subsequently with three sets. In each of the experiments we found that the top ranked motifs were exactly as reported in Figures 2 to 5.

Furthermore we compared the motifs extracted by Varun with those obtained by MEME [6]. MEME is an enumerative approach that uses expectation maximization to extract patterns that are over-represented with respect to an expected frequency. For each family we consider the top 5 motifs obtained with MEME and among these 5 motifs we choose the one that overlaps the most with the region of the functionally relevant motif for that family. Figure 6 displays a summary of the findings, along with the number of sequences covered by the motifs. In most cases MEME was not able to detect the correct motif or a portion of it. Only in one case MEME discovered a motif similar to the one discovered by Varun, but with a smaller coverage.

Protein Family and Motifs	Coverage
Streptomyces Subtilisin-type Inhibitors Signature	
RA.T[LV].C.P-(2,3)G.HP....AC[ATD].L....[ASG]	20/20
RAVTL[TSN]CAP[TG][AP]SGTHPA[AP]A[AS]ACAELR	15/20
Nickel-dependent Hydrogenases	
[LIM]-(1,2,3,4)[STA][FY]DPC[LIM][ASG]C[ASG].H	22/22
–	0/22
G-protein coupled receptors	
CC[FYW].C..C....[FYW]-(2,4)[DN]..[STAH]C..C	25/25
–	0/25
Coagulation factors	
P-(4,5,8,9,10)[LM]R.[GE][LIVP].GC	40/40
–	0/40

Fig. 6. Comparison of the best motifs discovered by Varun (in bold) and by MEME.

Markov Model. The experiments in this set were carried out assuming that the input sequence is generated by a first-order Markov process. The transition and stationary probabilities of the model are estimated from the union of all sequences under study.

Under the Markov model, the probability $p(m)$ of a motif is smaller than the one computed assuming an iid source. However, the relative rankings of the motifs is more or less unchanged in the two models. Consistently, the functionally relevant motifs still appear in the first position (see Figure 7). Due to space constraints we report only a small selection of experiments, some more are included in the supplemental material.

Rank	z-score	Motif
1	**3,93205e+07**	**RA.T[LV].C.P-(2,3)G.HP....AC[ATD].L....[ASG]**
2	3416,13	A..[LV].C.P-(2,3)G..P-(1,2,4)A.[ATD]
3	1937,72	P-(1,2,3)[ASG]..P-(2,3,4)AC[ATD].L....[ASG]
4	429,241	A-(1,4)T[LV].C.P-(1,2,3)G
5	381,427	R..T....P-(2,3)[ASG]..P-(1,2,4)[ASG].[ATD]
6	356,925	P-(1,2,3)[ASG]..P....[ATD].[ATD].[LV]-(1,2,3,4)G
7	336,666	P.[ASG]-(2,3,4)P....[ATD].[ATD].L....[ASG]
8	303,414	P.[ASG]-(2,3,4)P....[ASG].[ATD].L....[ASG]
9	303,199	R[ASG].[ATD][LV].C-(1,3)P

Fig. 7. (MARKOV) The functionally relevant motif is shown in bold for Streptomyces subtilisin-type inhibitors signature (id PS00999). Here 20 sequences of about 2500 amino acids were analyzed at k=20, D=4, d=4.

Rigid motifs. By an appropriate setting of parameters, Varun will extract rigid motifs. We report here two examples from the same data set and under iid assumptions. Figure 8 shows how Varun can capture the rigid parts of target motif. In particular, the

program extracts correctly the two rigid parts of the Streptomyces subtilisin-type inhibitor motifs. Similar results can be found in the supplemental material.

Rank	z-score	Motif
1	**15198,2**	**G.HP....AC[ATD].L....[ASG]**
2	**1864,51**	**RA.T[LV].C.P**
3	188,305	G..P....A...L....[ASG]
4	150,247	R[ASG].[ATD][LV].C
5	131,917	F[ASG]N.C
6	128,104	[ASG]..P....A...L....[ASG]
7	118,452	[ATD].T..C.P
8	79,2423	R[ATD].[ATD]..C
9	70,1957	R..T....P...[ASG]
10	67,2725	R[ASG]..[LV].C
11	64,7149	P....A...L....[ASG]
12	59,1431	A.[ATD][LV].C

Fig. 8. Rigid motifs for Streptomyces subtilisin-type inhibitors signature (id PS00999). Here 20 sequences of about 2500 amino acids were analyzed at k=20, D=4, d=4.

Experiments with DNA. We also tested the performance of Varun on DNA sequences, namely, on the upstream regions of co-regulated genes. SCPD [20] is a database of different transcription factors for Yeast. For each transcription factor, the published motif pattern, the positions of the binding sites and the set of sequences containing these binding sites are recorded.

We tested the ability of Varun, MEME [6] and Projection [9] in discovering the published motif for each data set. For MEME we choose the motif that overlap the most with the solution among the top 5 motifs reported. Since Projection required the motif length l and Hamming distance d as input parameters, we set l to be the length of the published motif and tested all values of d from 0 to l. That algorithm is said to discover a published motif if it outputs it for some d. On the other hand, Varun does not require to know the length of the motif in advance, and the only parameter that needs to be set is D, that in the examples reported is particularly small, namely, $0 \leq D \leq 2$.

Figure 9 shows the results of these experiments. Even though we did not input the lengths of all of them, Varun could discover all the published motifs. Moreover in some cases (GAL4, ROX1 and UASGABA) Varun could discover the correct motifs while Projection could not. Similar considerations are valid also for MEME that was not able to find two motifs. Figures 10-12 present a detailed summary of the experiments reported in Figure 9. In almost all cases the published motif is recovered correctly, typically with a more specific version, and it appears in one of the top positions. Only in Figure 11 a number of non significant motifs appear in the top few positions. This is due to the sequences involved, particularly rich of runs of Ts. Such a situation is easy to recognize and filter out. As for the experiments on protein data the z-score appears as a good indicator for the quality of the motifs extracted.

Rank	z-score	Motif
1	**75,9558**	**CGG.G..CT.T.G..CG**
2	47,351	A..AA.A..AA.A.TT
3	47,302	AA.A..AT.A..AAA
4	38,0177	C..A..A..AA..T.A.A.A
5	38,0177	CA.A..A..A..A..A.A.A
6	38,0177	T..C..TTT.T..T..T..T
7	37,978	G.A..G..G.T..TT..TC
8	37,9385	TTT..GG..A..A..A.G
9	37,899	A..A..AA.AT..TA.A
10	37,899	GG..G..T.T..TCC.T
11	37,8597	A..A..AG.AA.A.TT
12	37,8597	AA..TTT.C..TT..T
13	37,7815	C.T.ATAA..A.AA
14	37,7815	CT..TC..CCG..CG
15	32,9745	A..A..AT.A..AAA
16	32,906	A..AT.A..AAAA

Fig. 10. Motifs extracted from DNA sequences of the transcriptional factor: GAL4. Parameters used: D=2, k=4.

Rank	z-score	Motif
1	391,67	TTTTTTTTT
2	261,94	TCCTTTTTTTTT
3	244,46	TTTTTTTT
4	196,25	CCTTTTTTTTT
5	150,43	TTTTTTT
6	**130,83**	**CCATTGTTCTC**
7	99,19	TTTTTT
8	98,00	TTTTTTTTTC
9	81,55	TTTTTTTTC
10	65,32	ATTGTTCTCG
11	65,32	TTTTTTTTCC
12	62,04	TTTTT
13	48,90	CCATTGTTC
14	48,90	ATTGTTCTC

Fig. 11. Motifs extracted from DNA sequences of the transcriptional factor: ROX1. Parameter used: D=0, k=2.

Rank	z-score	Motif
1	**8469,49**	**G.CAAAA.CCGC.GGCGG.A.T**
2	1056,48	A.CGC.GCTT.G.AC.G.AA
3	528,79	GG.A.TC.T.T.G.TA.T.GC
4	527,143	TT.GA.ATG.TTT.T.TC
5	263,566	GT.CG.T.AT.G.ATA.G
6	263,293	TT.TC.T.C.CC.AAAA
7	263,293	GAT.ATA.AA.A.AG.A
8	263,293	CA.A.TA.TCA.TT.CT
9	263,293	T.TA.G.T.TTT.CTTC
10	263,022	T.ATA.T.TATTAT.A
11	131,499	ATA.A.AA.AG.A.AA
12	131,499	T.TTT.CTT.T.CC.A
13	131,364	G.TGT.AT.AT.TAA
14	131,229	C.T.AATAA.AAAT
15	131,229	TAT.G.TAATC.CT

Fig. 12. Motifs extracted from DNA sequences of the transcriptional factor: UASGABA. Parameters used: D=1, k=2.

V. Conclusion & Future Directions

The case studies reported in Section IV support the view that allowing extensibility in a motif not only leads to a succinct description but also helps in capturing function and/or structure in a single pattern. On the other hand, with extensible motifs the number of candidates to be considered increases dramatically. By rigidly conjugating structure and set of occurrences, our characterization of an extensible motif pattern lends itself to a natural notion of maximality. Such a notion supports in turn a discovery process in which motifs that are unlikely to be surprising can be discarded before their score is computed. The viability of the method has been tested successfully on families of proteins and DNA sequences. More advanced probabilistic frameworks seem worthy of investigation.

Acknowledgments

Work by A. Apostolico was supported in part by the Italian Ministry of University and Research under the National Projects FIRB RBNE01KNFP and PRIN "Combinatorial and Algorithmic Methods for Pattern Discovery in Biosequences", and the Bi-National Project FIRB RBIN04BYZ7; by the Research Program

Name	Motif pattern	$PROJECTION$	$MEME$	$Varun$	Rank(z-score)
CuRE	TTTGCTC	TTTGCTC	–	TTTGCTCA	1 (24,3356)
GATA	CTTATC	CTTATC	GCTTATC	GCTTATC	1 (17,64)
GAL4	$CGG.....CCG$	–	–	CGG.G..CT.T.G..CG	1 (75,9558)
ROX1	[CT][CT].ATTGTT[CT]	–	CC[CT]ATTGTTCTCG	CCATTGTTCTC	6 (130,83)
UASGABA	AAAAACCGCCGGCGGCAAT	–	CCGCC[CG]GCGGC	G.CAAAA.CCGC.GGCGG.A.T	1 (8469,49)

Fig. 9. Comparison of different methods on DNA: transcription factors for Yeast.

of the University of Padova, and by Georgia Tech Research Foundation. Work by M. Comin was performed in part during his internship at IBM Thomas J. Watson Research Center. We thank the reviewers for their valuable comments.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.

[2] A. Apostolico, M. E. Bock, and S. Lonardi. Monotony of Surprise and Large Scale Quest for Unusual Words. *Journal of Computational Biology*, 10(3-4), pp. 283-311 (2003).

[3] A. Apostolico, M. Comin, and L. Parida. Conservative Extraction of Over-represented Extensible Motifs. In Proc. International conference on Intelligent Systems for Molecular Biology (ISMB 2005), *Bioinformatics*:21 Suppl 1, pp. 9-18, 2005.

[4] A. Apostolico and L. Parida. Incremental paradigms for motif discovery. *Journal of Computational Biology*, 11(1):15–25, 2004.

[5] A. Apostolico and C. Pizzi. Monotone scoring of patterns with mismatches. In *Proc. of WABI*, volume 3240, pages 87–98, September 17-21, 2004.

[6] T. Bailey, N. Williams, C. Misleh, and W Li. MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Research*, Vol. 34, pp. W369-W373, 2006.

[7] A. Blumer, J. Blumer, Ehrenfeucht A., D. Haussler, M.T. Chen, and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science*, 40:31–55, 1985.

[8] D. R. Breiter, T. E. Meyer, I. Rayment, and H. M. Holden. *J. Bio. Chem.*, 266:18660–18667, 1991.

[9] J.. Buhler and M. Tompa. Finding Motifs using Random Projections. In *Journal of Computational Biology*, 9:2, pp 225–242, 2002.

[10] A. Chattaraj and L. Parida. An inexact suffix tree based algorithm for extensible pattern discovery. *Theoretical Computer Science*, 335(1):3-14, 2005.

[11] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15:563–577, 1999.

[12] U. Keich and P.A. Pevzner. Finding motifs in the twilight zone. In *Annual International Conference on Computational Molecular Biology*, pages 195–204, Apr, 2002.

[13] C. E. Lawrence, Altschul, Boguski S. F., J. S. M. S., Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, Oct, 1993.

[14] M. Y. Leung, G. M. Marsh, and T. P. Speed. Over and underrepresentation of short DNA words in herpesvirus genomes. *Journal of Computational Biology*, 3:345–360, 1996.

[15] L. Parida. Pattern Discovery in Bioinformatics: Theory and Algorithms. Mathematical and Computaional Biology Series, Chapman Hall/CRC, 2007.

[16] P. A. Pevzner and S.-H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press, 2000.

[17] I. Rigoutsos and A. Floratos. Motif Discovery In Biological Sequences Without Alignment Or Enumeration. In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, pages 221–227, 1998.

[18] J.T.L. Wang, B.A. Shapiro, D. Shasha. *Pattern Discovery in biomolecular Data*. Oxford University Press, 1999.

[19] M.S. Waterman. *An Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman Hall, 1995.

[20] J. Zhu, and M. Zhang. SCPD: a promoter database of the yeast Saccharomyces cerevisiae. *Bioinformatics*, Vol. 15, pp.563-577.

Alberto Apostolico Alberto Apostolico is Professor of Computer Engineering at the University of Padova and Professor of Computing at Georgia Institute of Technology. His research interests are in the area of algorithm design and analysis and its applications. Most of his work deals with algorithms and data structures for combinatorial pattern matching and discovery problems as arising in biomolecular sequence analysis and other domains.

Matteo Comin Matteo Comin received the laurea degree in computer science in 2003 from the University of Padova (Italy). In 2003 he was visiting student at Purdue University, in 2004 and 2006 research intern at the IBM T.J. Watson Research Center. In 2007 he received a PhD in computer science from the University of Padova. Currently, he is an assistant professor at the same University. His interests are in the design of algorithms and applications for pattern discovery, data compression, geometric pattern matching, structural proteomics.

Laxmi Parida Laxmi Parida is a research staff member in the Computational Biology Center, at the IBM T.J. Watson Research Center, Yorktown Heights and a visiting professor at New York University. She obtained her PhD in Computer Science, at the Courant Institute of Mathematical Sciences, 1998, in the area of computational genomics. Her research focuses on problems arising in the general area of computational biology and bioinformatics.