

# SPLASH: Structural Pattern Localization Analysis by Sequential Histograms

*Andrea Califano<sup>1</sup>*

*IBM TJ Watson Research Center  
PO Box 704, Yorktown Heights, NY 10598*

## Abstract

Pattern discovery, usually, is either reduced to an enumeration and verification problem or to a multiple alignment problem. Either class of problems has been shown to be NP-Hard [8, 9]. Therefore, most of the solutions that have been proposed use heuristics or ad hoc constraints to discover patterns efficiently. Probabilistic algorithms such as Meme [11], for instance, maximize a likelihood function. Enumeration algorithms, such as Pratt [10], limit the maximum size of discovered patterns to avoid exponential requirements on system memory. A more recent approach, used by the Teiresias [12] algorithm, is to enumerate only patterns that are consistent with the dataset. This is accomplished by combining smaller patterns that have an identical prefix and suffix to form increasingly larger patterns. This assumes patterns over a fixed alphabet  $\Sigma$  that repeat exactly.

This paper introduces a novel pattern discovery algorithm, called Splash. The relevance of this algorithm is threefold. First, the signal is not limited to a fixed alphabet size and the patterns do not have to match identically. Rather, they are modeled by a homology metric. Second, it achieves significantly higher computational efficiency by early pruning of inconsistent seed patterns. Third, Splash is embarrassingly parallel in nature and scales linearly with hardware resources.

The need to discover homologous patterns should be self evident. Most biological patterns, including those that arise in protein sequence analysis, protein motif discovery, and gene expression analysis cannot be modeled by an exact match over a fixed size alphabet. Many amino acids, for instance, are easily interchanged by evolution without loss of function [7]. All existing pairwise sequence matching algorithms, such as BLAST [13], FASTA [16], and Smith-Waterman [17], rely on amino acid distance matrices, such as PAM [3] or BLOSUM [14], to increase the chance of a match. Similarly, gene expression values are continuous variables that are best compared using a euclidean or Hamming distance metric. In all these cases, as it will be shown in this paper, exact match patterns imply a significant loss of sensitivity. Experiments on the histone and GPCR super family are reported.

## 1. Introduction

Whenever Nature finds a “recipe” to accomplish a task that gives a differential fitness to an organism, chances are that such recipe will be conserved through evolution. At the molecular level, this means that biological sequences, belonging sometimes to widely distant species, will share common motifs. Thus, the identification of patterns in biological databases is becoming a very transited venue within the bioinformatics community. Pattern databases such as PROSITE [2] are examples of this trend.

In recent years a number of interesting pattern discovery tools have emerged (Teiresias [12], Pratt [10], Meme [11], among others) to help automate the task of motif discovery. These algorithms are

---

1. The results in the “Pattern Relevance” section were obtained in collaboration with Gustavo Stolovitzky. They will appear independently in a separate joint publication.

valuable in that they can discover weak motifs, hidden in the biosequences. A discussion of some of the most interesting ones is available in [9]. A statistical framework [4] has also been proposed to help determine the statistical relevance of discovered patterns.

Due to the combinatorial nature of pattern discovery, however, these algorithms are either of a probabilistic nature, or impose severe limitations on the variety of discovered patterns. Meme, for instance, uses a likelihood function to avoid looking for patterns in low probability regions. Teiresias produces patterns that match identically on a fixed size alphabet.

None of the existing algorithms offer a deterministic and efficient solution to the problem of homologous pattern discovery. This paper introduces a new algorithm called Splash that offers an efficient, embarrassingly parallel solution to the problem of patterns discovery in either discrete or continuous data sets where the patterns determine arbitrary homology classes. One such class is the identity class where patterns match only when they appear identically.

Many biological problems can be best modeled using similarity metrics rather than identity. Among others, Splash can be applied to the problem of determining patterns in 3-dimensional arrangements of amino acids, to the problem of determining amino acid patterns in protein databases when their compatibility is described by a probability of mutation matrix [3], and to the problem of detecting patterns in gene expression arrays with arbitrary tolerances for the match on the expression level range [5].

This paper reports some preliminary results for the Histone family and for the G-protein-coupled receptor superfamily as well as theoretical and experimental results that show the increase in sensitivity deriving from an appropriate selection of the homology class mechanism, such as for instance the one determined by PAM or BLOSUM matrices.

Splash is both computationally and memory efficient. For instance, the full non-redundant PDB [18] can be processed in about 1 hour on a 266MHz Pentium laptop with 128MB of memory to discover all patterns that occur at least 3 times and have at least 7 homologous residues in a window of 12. The algorithm is embarrassingly parallel in nature and it has been implemented as such to run on RS/6000 SMP and SP architectures. Parallel Splash can analyze a non-redundant protein database, with over 90 million residues, in about 10 hours on a 4-way SMP workstation using about 300MB per process to discover any pattern that has at least 6 matching characters in a window of 10 characters that would not occur randomly in a database of that size. Typical runs on large protein families such as the GPCR or the Histones can be performed in seconds to minutes on Pentium laptops and workstations. A Java client-server version of Splash will be available on the Web early in 1999 and a version for any of the Windows OS is freely available to research institutions by signing a field test agreement.

## 2. Definitions and Notation

Let us denote by  $s = \{v_\omega | \omega = 1, \dots, L\}$  a string of  $L$  values. The values  $v_\omega$  (that we shall also call tokens) are taken from a given alphabet  $\Sigma$ , which in concrete examples can be the four bases A,C,G,T or the twenty amino acids. More generally,  $\Sigma$  could be a continuous set, such as an interval on the real line. The composition of a pattern  $\pi$  is a set of (value, relative offset) pairs  $\eta_\pi = \{(v_i, \delta_i) | i = 1, \dots, k\}$ . Usually  $\delta_1 = 0$ . A pattern with composition of cardinality  $k$  is called a  $k$ -pattern. A  $k$ -pattern whose span is  $l$  (i.e., the last relative offset is  $\delta_k = l - 1$ ) will be called a  $kl$ -pattern.

The locus  $\Omega_\pi = \{\omega_1, \omega_2, \dots, \omega_j\}$  of a given pattern  $\pi$  is the set of  $j$  absolute offsets where  $\pi$  occurs in  $s$ . We will define what we mean by "occur" in the following subsection. A  $kl$ -pattern that occurs at  $j$  different absolute offsets in  $s$ , i.e., has a locus of cardinality  $j$ , is called a  $jkl$ -pattern.

Let us define the translation operator for a locus  $\Omega_\pi + \tau = \{\omega_1 + \tau, \omega_2 + \tau, \dots, \omega_j + \tau\}$  and for a composition  $\eta_\pi + \tau = \{(v_i, \delta_i - \tau)\}$ . A pattern is translated by  $\tau$  if both its locus and its composition are translated by  $\tau$ .

Then, we say that a pattern  $\pi_b$  is a sub-pattern of another pattern  $\pi_a$  with span  $l_a$ , or that  $\pi_a$  contains  $\pi_b$ , if there exists a translation  $0 \leq \tau \leq l_a$  such that  $\Omega(\pi_a) \supseteq \Omega(\pi_b) + \tau$  and  $\eta_{\pi_b} \supset \eta_{\pi_a} + \tau$ . We also say that  $\pi_a$  is a super-pattern of  $\pi_b$  if  $\pi_b$  is a sub-pattern of  $\pi_a$ . If  $\tau = 0$  we will call  $\pi_a$  an aligned super-pattern of  $\pi_b$ .

Let us define the *addition* operator  $\pi_c = \pi_a + \pi_b$ . This generates a pattern that has a composition equal to the union of the added compositions  $\eta_c = \eta_a \cup \eta_b$ , and locus equal to the intersection of the added loci  $\Omega(\pi_c) = \Omega(\pi_a) \cap \Omega(\pi_b)$ . If  $\pi_b$  is a sub-pattern of  $\pi_a$  with translation  $\tau$ , it is sometimes convenient to add  $\pi_a$  and  $\pi_b + \tau$  to form a longer pattern with the full locus of  $\pi_b$ .

The set of  $k$  relative offsets in a  $k$ -pattern  $\psi = \{\delta_i | \delta_1 = 0; i = 1, \dots, k\}$  will be called a  $k$ -*comb*. If the span of a  $k$ -comb is  $l$ , where  $l = \delta_k + 1$ , the comb is called a  $kl$ -comb. It is sometimes convenient to represent a  $kl$ -comb with a string of 0s and 1s that has a 1 at each relative offset  $\delta_i$  and a 0 elsewhere. Then, for instance, the comb  $\psi_5 = \{0, 3, 4, 7, 11\}$  can be represented by the string 100110010001. By  $\pi_\psi$  we shall indicate a pattern supported by the comb  $\psi$ . By  $\pi_{\psi, \omega}$  we shall indicate the pattern determined by the comb  $\psi$  at offset  $\omega$  in  $s$ .

A convenient notation for patterns is as a one dimensional string containing tokens and the wild character “.”. This is a subset of typical regular expressions. For instance, for an alphabet  $\Sigma = \{A, C, G, T\}$ , the pattern  $\pi = \{(A, 0), (C, 3), (G, 7)\}$  is a  $kl$ -pattern, with  $k = 3$  and  $l = 8$  which can be written as  $\pi = \mathbf{A} . . \mathbf{C} . . . \mathbf{G}$ . If the pattern can match any of a number of characters at a given offset, we will use the standard regular expression notation  $[v_1 v_2 \dots v_m]$ . For instance a pattern matching either  $\mathbf{A} . \mathbf{B} . \mathbf{C}$  or  $\mathbf{A} . \mathbf{D} . \mathbf{C}$  will be denoted as  $\mathbf{A} . [\mathbf{BD}] . \mathbf{C}$ .

## 2.1 Homology Metrics

Since we are interested in patterns that define a homology class, let us define a homology metric  $H(x, y)$  as a positive function of  $x$  and  $y$ , such that  $H(x, x) = 0$  and which satisfies the triangle inequality:

$$H(x, y) \leq H(x, w) + H(w, y) \quad (2.1)$$

A symmetric homology metric, i.e.,  $H(x, y) = H(y, x)$ , is equivalent to a distance metric [19].

For instance, if the log-probabilities of amino acid mutation are used to define homology classes,

$$H(x, y) = -\log(p_{x \rightarrow y}), \quad (2.2)$$

the resulting metric will be slightly asymmetric. For instance, the probability of alanine to mutate into aspartic acid is slightly larger than the probability of aspartic acid to mutate into alanine. Therefore,  $H(\mathbf{Ala}, \mathbf{Asp}) < H(\mathbf{Asp}, \mathbf{Ala})$ . However, the triangle inequality is still satisfied because  $p_{x \rightarrow y} \geq p_{x \rightarrow w} \cdot p_{w \rightarrow y}$  and therefore,  $\log(p_{x \rightarrow y}) \leq \log(p_{x \rightarrow w}) + \log(p_{w \rightarrow y})$ .

Given a homology metric  $H$  and a threshold  $H_0$ , we shall say that a  $k$ -pattern matches  $s$  at offset  $\omega$  or, conversely, that it occurs in  $s$  at offset  $\omega$  if and only if the relation  $H(v_i, v_{\omega + \delta_i}) \leq H_0$  holds true for each value of  $1 \leq i \leq k$ .

## 2.2 Homology Metrics for Protein Sequences

As mentioned above, a convenient homology metric for protein sequence analysis can be defined using the logP of the amino acid mutation probabilities, such as the ones in PAM or BLOSUM matrices. For instance, using a BLOSUM50 mutation probability matrix [14], and a threshold  $H_0 = 2$ , the amino acids define the following homology classes:

$$\begin{array}{lll}
\mathbf{Ala} \approx [\mathbf{Ala}] & \mathbf{Arg} \approx [\mathbf{Arg}, \mathbf{Lys}] & \mathbf{Asn} \approx [\mathbf{Asn}, \mathbf{Asp}] \\
\mathbf{Asp} \approx [\mathbf{Asp}, \mathbf{Glu}] & \mathbf{Cys} \approx [\mathbf{Cys}] & \mathbf{Gln} \approx [\mathbf{Gln}, \mathbf{Glu}, \mathbf{Lys}] \\
\mathbf{Glu} \approx [\mathbf{Glu}, \mathbf{Asp}, \mathbf{Gln}] & \mathbf{Gly} \approx [\mathbf{Gly}] & \mathbf{His} \approx [\mathbf{His}, \mathbf{Tyr}] \\
\mathbf{Ile} \approx [\mathbf{Ile}, \mathbf{Leu}, \mathbf{Met}, \mathbf{Val}] & \mathbf{Leu} \approx [\mathbf{Leu}, \mathbf{Ile}, \mathbf{Met}] & \mathbf{Lys} \approx [\mathbf{Lys}, \mathbf{Arg}, \mathbf{Gln}] \\
\mathbf{Met} \approx [\mathbf{Met}, \mathbf{Ile}, \mathbf{Leu}] & \mathbf{Phe} \approx [\mathbf{Phe}, \mathbf{Tyr}] & \mathbf{Pro} \approx [\mathbf{Pro}] \\
\mathbf{Ser} \approx [\mathbf{Ser}, \mathbf{Thr}] & \mathbf{Thr} \approx [\mathbf{Thr}, \mathbf{Ser}] & \mathbf{Trp} \approx [\mathbf{Trp}, \mathbf{Tyr}] \\
\mathbf{Tyr} \approx [\mathbf{Tyr}, \mathbf{His}, \mathbf{Phe}, \mathbf{Trp}] & \mathbf{Val} \approx [\mathbf{Val}, \mathbf{Ile}] & 
\end{array} \quad (2.3)$$

Then, given the sequence

$$\mathbf{Ala} \text{ Cys Gln Gln } \mathbf{Val} \mathbf{Trp} \text{ Ala Gly Ala Phe Ile Tyr Leu His Pro} \quad (2.4)$$

the pattern  $\{(\mathbf{Ala}, 0)(\mathbf{Ile}, 4)(\mathbf{Tyr}, 5)\}$ , for instance, would have locus  $\lambda = \{0, 6, 8\}$ . Note, however that, based on the same metric, the pattern  $\{(\mathbf{Ala}, 0)(\mathbf{Val}, 4)(\mathbf{Trp}, 5)\}$ , which appears at position 0, would not have the same locus since both  $f(\mathbf{Val}, \mathbf{Leu}) > 2$  and  $f(\mathbf{Trp}, \mathbf{His}) > 2$ .

### 2.3 Maximality

A  $jkl$ -pattern is said to be *maximal in composition* if it cannot be extended to a  $j(k+1)l$ -pattern, by adding an extra token within its span, without simultaneously decreasing the cardinality of its locus. A  $jkl$ -pattern is said to be *maximal in length* if it cannot be extended to a  $j(k+1)l'$ -pattern (with  $l' > l$ ), by adding an extra token outside its original span  $l$ , without simultaneously decreasing the cardinality of its locus. A pattern that is maximal in length on its right side is called right-maximal. Conversely, a pattern that is maximal in length on the left side is called left-maximal. A pattern that is both left- and right-maximal is maximal in length.

A *maximal pattern* is a pattern that is both maximal in composition and in length. Maximality is an essential property of pattern discovery algorithms. It avoids reporting a combinatorial number of sub-patterns of each maximal pattern in the sequence.

### 2.4 Density Constraint

It is also important to be able to specify the minimum number of tokens allowed in any interval on a pattern  $\{[\delta_i, \delta_i + l_0] | i = 1, \dots, k\}$  that starts at one of the relative offsets of the pattern. If such constraint is left unspecified, any  $j$ -pattern will grow indefinitely towards an average length

$$\bar{k} = L \sum_{\Sigma} p(v_i) \left[ \sum_{\Sigma} p(v_j) \alpha_{ij} \right]^{j-1} \quad (2.5)$$

where  $[p(v)]^{-1}$  is the frequency of the symbol  $v$  in  $s$  and  $\alpha_{ij}$  is one if  $H(v_i, v_j) \geq H_0$  and zero otherwise. If  $v$  is a continuous variable, the sum must be replaced by an integral.

Since  $d = k_0/l_0$  can be interpreted as a token density, we shall refer to such constraint as the *density constraint*. Given  $k_0$  and  $l_0$ , we shall say that a comb is a  $\langle k_0, l_0 \rangle$ -valid comb, if the relation  $\delta_{i+k_0-1} - \delta_i \leq l_0$  is satisfied for all  $1 \leq i \leq k - k_0 + 1$ . A pattern  $\pi_{\Psi}$ , supported by a  $\langle k_0, l_0 \rangle$ -valid comb is a  $\langle k_0, l_0 \rangle$ -valid pattern.

## 3. The Algorithm

Details about the algorithm are still under IP review and will be provided in a follow-up paper. However, an implementation of the algorithm will be available on the world-wide-web by time of publication. Non-profit scientific organization and research centers can obtain a working copy of the executable by requesting a field test agreement.

An important feature of Splash is its embarrassingly parallel nature. This is because the algorithm does not require access to previously discovered patterns to verify the maximality of a newly discovered pattern. As a result, Splash can be run in parallel on low-bandwidth distributed architectures. The only inter-process communication requirement is that the data is available on each processor and that the final results are collected at the end.

Its computational efficiency is mostly the result of the early pruning of potential pattern candidates, before they have a chance to contribute to an exponential growth of hypotheses.

## 4. Pattern Relevance

In [4], it is shown how the average number of maximal  $jkl$ -patterns in a random database  $s$  of length  $L$ , with a  $\langle l_0, k_0 \rangle$  density constraint, is given by

$$\langle v_{jkl} \rangle \approx N_0(k, l) \binom{L-l+2}{j} \langle \wp \rangle^{k(j-1)} [1 - \langle \wp \rangle^k]^{L-l-2-j} \langle p_{in} \rangle \langle p_{out} \rangle^2 \quad (4.1)$$

In the above expression,  $N_0(k, l)$  is the number of  $\langle l_0, k_0 \rangle$ -valid combs that have a span  $l$ , and  $k$  tokens.  $\langle \wp \rangle$  is given by the following expression,

$$\langle \wp \rangle = \sum_{\mathfrak{v}} p(\mathfrak{v}) [\wp(\mathfrak{v})], \quad (4.2)$$

where  $p(\mathfrak{v})$  is the probability of the value  $\mathfrak{v}$  to occur in the sequence and

$$\wp(\mathfrak{v}) = \sum_{\mathfrak{v}'} p(\mathfrak{v}') H(\mathfrak{v}, \mathfrak{v}') \quad (4.3)$$

is the probability of the value  $\mathfrak{v}$  to randomly match any other character in  $s$ . Also,  $p_{in}$  and  $p_{out}$  are respectively the probability that a given  $jkl$ -pattern is maximal in composition and length. From this analysis, it is possible to estimate the probability that any discovered pattern would have occurred randomly. Details of this analysis are available in [4]. Splash uses the results of the analysis to compute a z-score  $Z_s$  for the reported patterns. This is defined as follows:

$$Z_s[\pi_{jkl}] = \frac{\mathfrak{v}(\pi_{jkl}) - \bar{\mathfrak{v}}(\pi_{jkl})}{\sigma(\pi_{jkl})}, \quad (4.4)$$

where  $\bar{\mathfrak{v}}(\pi_{jkl})$  and  $\sigma(\pi_{jkl})$  are, respectively, the mean and the standard deviation of the number  $jkl$ -patterns that would occur in a random database of that size and composition. The value of the variance is fully studied in [4].

## 5. Performance

To compare the performance of Splash to that of Pratt, a publicly available patted discovery algorithm, we have run two sets of tests. In the first, we are testing the algorithm against a synthetically randomized version of the Brookhaven sequence database using a support that is a fixed percentage (20%) of the number of sequences in the database. In the second, we are running against a non-redundant histone database with 209 proteins, at increasingly lower values of the support.

The random databases are generated by creating random sequences with exactly the same length and amino acid frequency of the model protein database. Total size of the databases is  $8192 \cdot 2^i$ , with  $0 \leq i \leq 6$ . The largest random database is approximately 500 kilo residues.

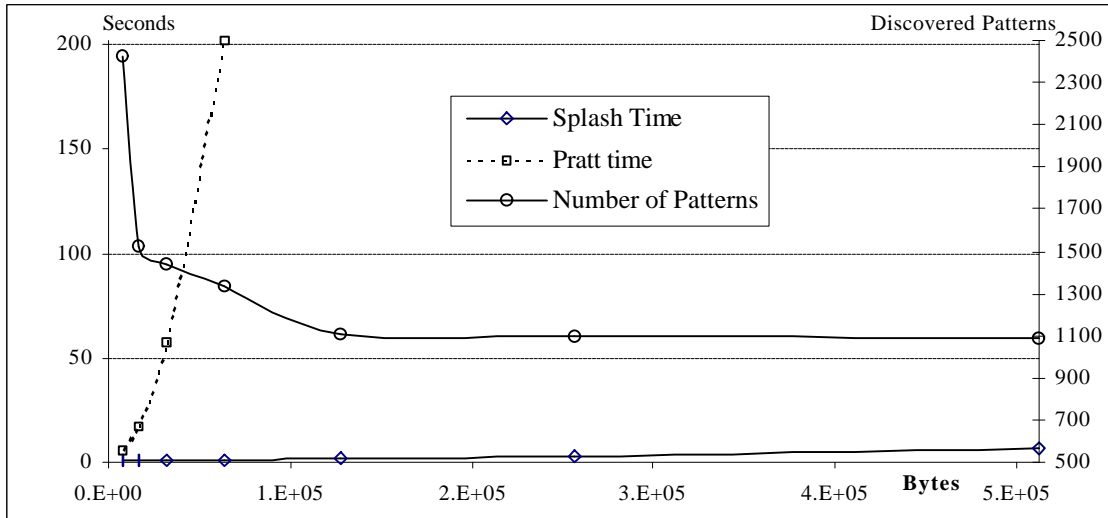


Fig. 1: Splash vs. Pratt. Time versus Database Size

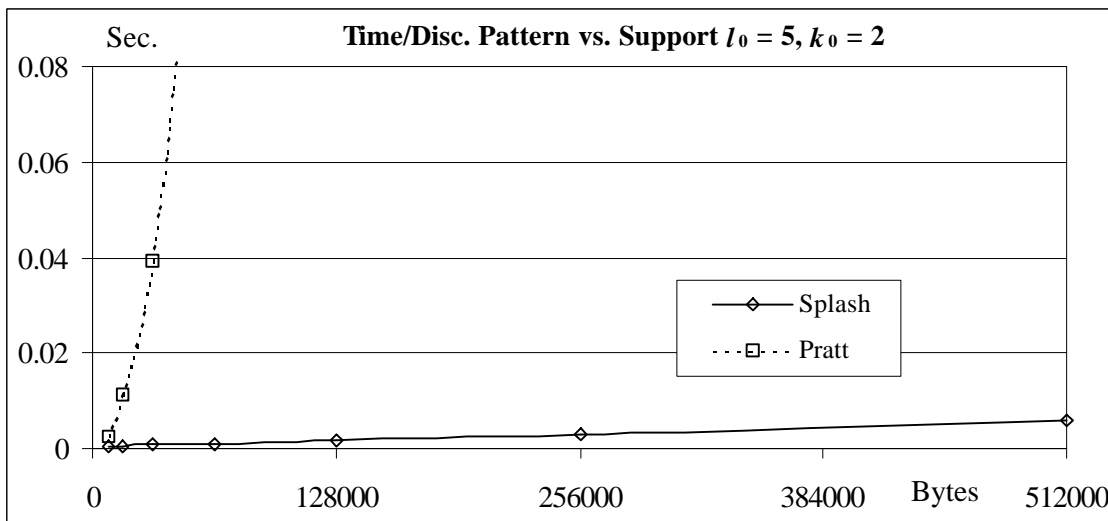


Fig. 2: Splash vs. Pratt. Time per discovered pattern.

In Fig. 1, we show the time in seconds required by Splash and Pratt to process a database, as a function of the length of the random databases, with  $i$  going from 0 to 6. The run parameters are:  $l_0 = 5$  and  $k_0 = 2$ , e.g., patterns must have at least two matching amino acids in any window of 5 amino acids. The minimum support is chosen to be 20% of the number of sequences in the databases. To make the comparison fair, we are running Splash in identity mode, i.e., only patterns that occur identically are discovered. The number of discovered patterns is identical for all algorithms.

In Fig. 1, we also plot the number of discovered patterns versus the database size against the right axis. Splash is between 6 times and several orders of magnitudes faster than Pratt on identical runs. The ratio increases monotonically as the database size also increase. This can be also seen in Fig. 2, where the time per discovered pattern is plotted. Besides the considerable difference, it should be noted that, in the case of Pratt, the time increases more than linearly with database size, while for Splash the increase is slightly sublinear and therefore more scalable.

In Fig. 3(a) and (b), we report the times for Splash in runs against the pattern rich histone database. This type of analysis can not be performed with Pratt, since there is no choice of parameters to report only patterns with a minimum number of tokens.

In Fig. 3(a), time is reported as a function of decreasing minimum support required. In Fig. 3(b) time is reported as a function of discovered patterns. Splash sublinear behavior is again very visible, especially when plotted against the dotted line that shows the linear extrapolation of Splash's performance for low number of patterns.

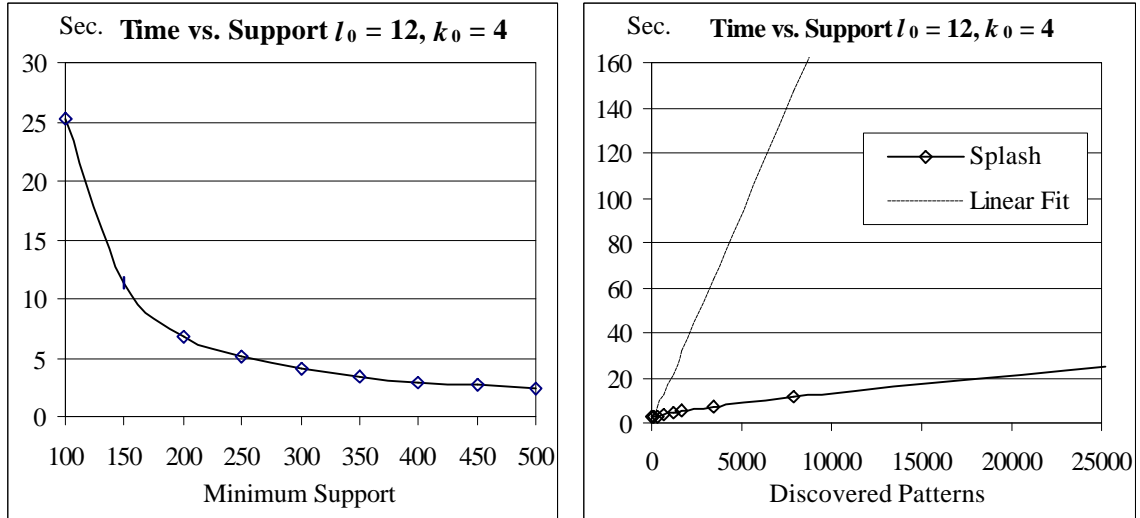


Fig. 3: Splash performance (a) vs. pattern support  $j$  (b) vs. the number of discovered patterns

## 6. Sensitivity of Identity vs. Homology Metrics

To show the increase in sensitivity obtained by using a homology metric instead of an identity one, we have performed the following analysis. First, a population of 10 random sequences of length 100, 200, 300, and 400 amino acid have been generated. The amino acid frequency has been chosen from the histones family. Subsequently, for each length, a random subsequence of 40 amino acid has been inserted identically in each one of the sequences. This subsequence has also been generated using the amino acid frequency of the histone family. From this initial population, mutated populations have been generated using the procedure described in [7], starting at an evolutionary period of 5 PAMs and increasing that in steps of 2 PAMs. Finally pattern analysis has been performed using Splash using first an identity and then an homology metric. The number of statistically significant patterns for the identity case has been plotted in Fig. 5(a) that for homology has been plotted in Fig. 5(b).

The goal of this exercise is to determine the likelihood of detecting a specific motif (chosen to be 40 residue in length), in protein families of different length, at increasing evolutionary intervals, through pattern discovery. That is, if any statistically significant pattern is discovered after  $x$  PAMs, we can say that pattern is what is left of an original motif and that its statistical significance is a measure of how related the sequences are.

The homology metric has been defined by BLOSUM50 with a threshold of 2, while the mutation rates have been produced by repeatedly applying PAM1 to the original sequences. We have deliberately used different matrices to show that even with different statistical approaches homologous patterns can better determine distant relationships. It is easy to see that the number of statistically significant patterns discovered with the identity metrics drops to zero quickly and that, with the exception of the shortest sequence length, which generates the smallest amount of noise (i.e., sta-

tistically irrelevant patterns), no patterns is reported after more than ten iteration with PAM1. On the other hand, the use of an homology metrics practically doubles the time horizon. the chances of missing the family relationship decrease very quickly as the evolutionary step is reduced as many more statistically significant patterns are reported.

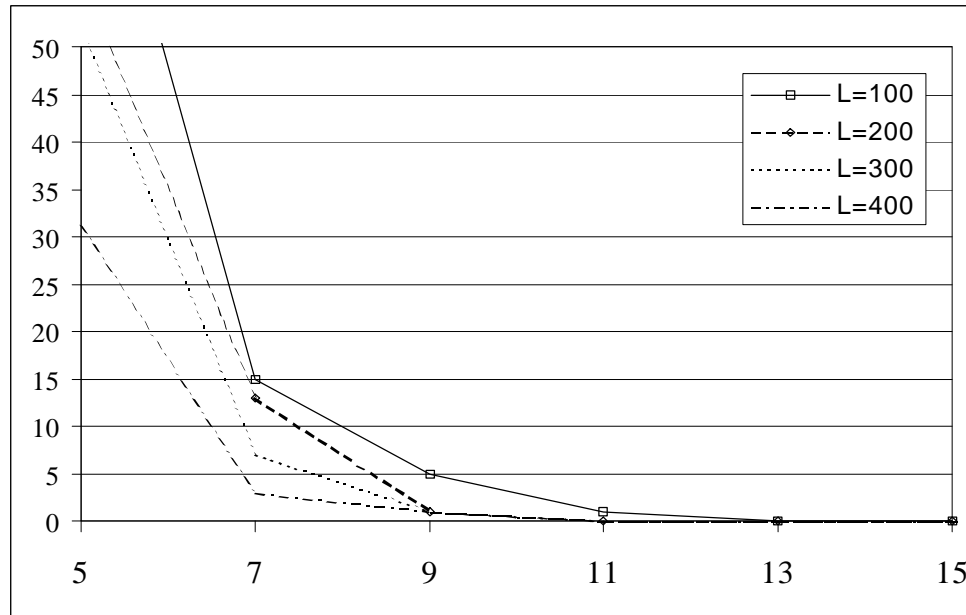


Fig. 4: Discovered patterns vs. evolutionary gap. Identity metric.

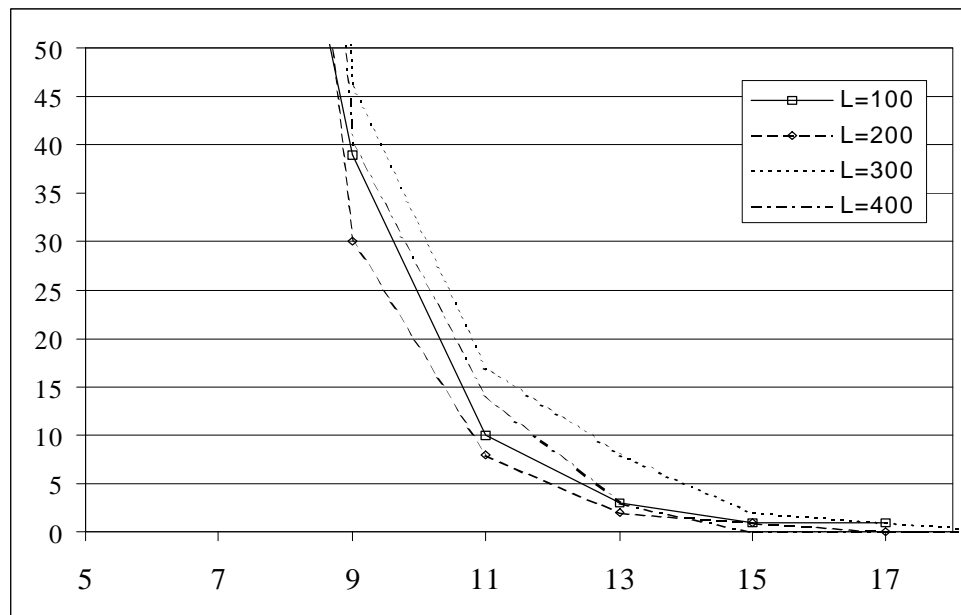


Fig. 5: Discovered patterns vs. evolutionary gap. BLOSUM50 homology metric

## 7. Experimental Results: Histones

An important measure of a pattern discovery algorithm's performance is its ability to discover hidden motifs in large protein databases. In this case, we have run splash against a non-redundant his-

tone H1 database with 209 proteins. If a homology metric defined by the BLOSUM50 matrix with a threshold of 2 is used and the density constraint has  $k_0 = 4$  and  $l_0 = 12$ , Splash detects the following motif:

$$\pi_0 = \mathbf{G.S...[ILMV]...[ILMV]}, (j = 209, Z_s = 3.013 \cdot 10^5)$$

Here  $v(\pi_{jkl}) = 1$  because we are considering each pattern independently. That is, the  $jkl$ -pattern occurred at least once.

Given the very large value of the  $Z_s$ , it is impossible for all practical purposes that a pattern like this would arise spontaneously from a database with a histone-like amino acid frequency of the same size. The analysis, accounts for the homologous matches on two of the four residues.

This is confirmed empirically by the analysis of the 3D structures of the globular domain of the only two histone proteins where the motif occurs in PDB. These are respectively the globular domain of the H1 [20] and H5 [21] histone from chicken. As can be seen from Fig. 6(a) and (b) where the two motifs are highlighted, there is significant 3D homology. The homology is the region between residues 22 and 32 on H1 and between residues 44 and 54 on H5.

The Glycine and Serine are exposed on the surface of the protein, in a loop at the end of an alpha-helix. The two Leucines in H1, mutated into Isolucines in H5, contribute to the stability of the hydrophobic core on the alpha-helix. Although the structural properties of this area of the histone H1 and H5 globular domain have not been fully understood, it could play an important structural role because it harbors the most conserved motif in that family.

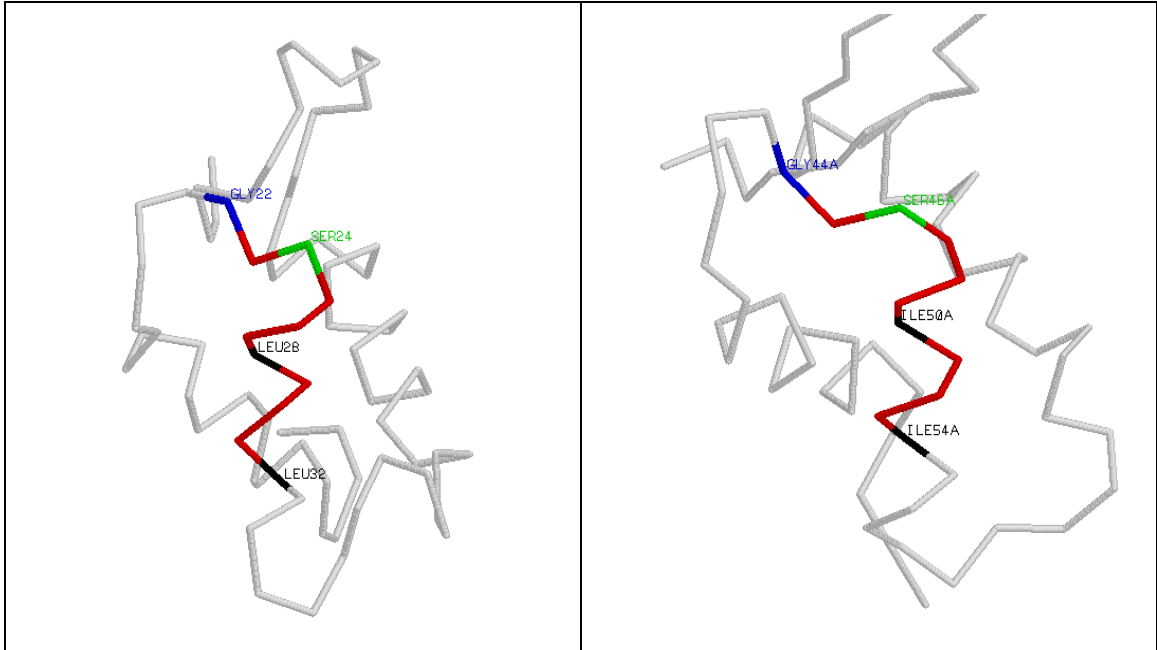


Fig. 6: Globular domain of histone proteins (a) Chicken histone H1 (b) Chicken histone H5

It should be noted that, in this case, the ability to discover patterns with a homology metric plays an important role. In particular, there are no 3 token sub-patterns of  $\pi_0$  that appear in all the sequences and sub-pattern  $\pi_3 = \mathbf{G.S...K}$ , which occurs in 201 proteins, has  $Z_s = -0.64$  which makes it virtually indistinguishable from background noise. In fact, there are about 170 patterns of this kind. From a statistical point of view, two character patterns, for this database, are completely irrelevant. Therefore, without the use of a homology metric, directly in the discovery process, pattern  $\pi_0$  would not readily emerge.

## 8. Experimental Results: G-Protein-Coupled Receptors

Another result is shown from a superfamily of 918 known and hypothetical G-Protein-Coupled Receptors. Here, Splash has been used to discover statistically relevant motifs that occur in the largest subset of proteins. If an exact match metric is used, even with a very low density constraint defined by  $k_0 = 4$  and  $l_0 = 30$ , a single 4 amino acid pattern is found that occurs in more than half of the proteins:

$$\pi_0 = \mathbf{N} \dots \mathbf{D R Y}, (j = 474, Z_s = -11.13)$$

This pattern contains the previously known and well preserved **DRY** tripeptide for family A. There Based on the  $Z_s$ , however,  $\pi_0$  is highly irrelevant in this context.

Under identical conditions, if the homology metric defined by the BLOSUM50 matrix with a threshold of 2 is used with the same density constraint, almost 500 patterns are discovered. These have at least 2 exact and 3 similar residues and occur in more than half of the proteins. The most frequent pattern with a  $Z_s \geq 3$  is

$$\pi_1 = \mathbf{C} \dots [\mathbf{I L M V}] \dots [\mathbf{I L M V}] \dots [\mathbf{N D E}] \mathbf{R} \dots [\mathbf{I L M V}], (j = 619, Z_s = 2.26 \cdot 10^{34})$$

This highly relevant pattern has 2 exact and 3 similar residue matches. This motif, used to search against Swissprot Rel. 36, returns 874 GPCR proteins and only 30 potential false positives, for a false/true ratio of 3.4%. About half of these are hypothetical or putative proteins. This is an improvement over the PROSITE motif **PS00237**, the most comprehensive one for GPCR proteins, which returns 841 correct positives and 44 false positives, for a false/true ratio of 5.2%. The marked improvement seems to be attributable to the Cystine in the first position in  $\pi_1$ .

Even a pattern with a significantly lower chance of appearing in a large database, such as

$$\pi_2 = \mathbf{C} \dots \mathbf{L} \dots [\mathbf{I L M V}] \dots [\mathbf{D E B}] [\mathbf{R Q K}] [\mathbf{H F W Y}] \dots [\mathbf{I L M V}], (j = 483, Z_s = 2.46 \cdot 10^{34})$$

with 2 exact and 5 similar amino acid matches, still occurs in more proteins than the exact pattern  $\pi_0$ , and it has a much higher  $Z_s$ . This motif is extremely selective. If it is used to search against swissprot Rel. 36, it returns 578 GPCR proteins and only 4 possible false positives. These are:

1. **C555\_METCA** Cytochrome C-555.
2. **KEMK\_MOUSE**: Putative Serine/Threonine-Protein Kinase EMK
3. **SRG8\_CAEEL**: SRG-8 Protein.
4. **UL29\_HCMVA**: Hypothetical Protein UL29.

Given the extremely low false positive ratio (0.68%), it is quite possible that either 2 or 4 may have been mis-labeled and should really be considered members of the GPCR family.

In this paper we limit ourselves to single pattern analysis. However, given the large number of statistically significant yet different patterns that can be discovered, even better classification results can be obtained by using multiple statistically significant patterns simultaneously. This will be the subject of a follow-up paper.

### 8.1 Acknowledgments

We would like to thank Gustavo Stolovitzky and Ajay Royyuru for their valuable help with this paper and for many useful suggestions. We would also like to thank Barry Robson, Aris Floratos, Laxmi Parida, Yuan Gao, Mike Pitman, and Isidore Rigoutsos for the many useful discussions on the topic of pattern discovery.

### 8.2 References

- [1] I. Rigoutsos, A. Floratos, and Christos Ouzounis: "Case Studies in Pattern Discovery without Alignment: Results Using the Teiresias Algorithm," IBM RC20803(92166), 4/26/1997

- [2] A. Bairoch. "PROSITE: a dictionary of sites and patterns in proteins." in *Nucleic Acids Res.* (1991) **19**, 2241-2245
- [3] R.M.Schwartz and M.O.Dayhoff: "Matrices for Detecting Distant Relationships," *Atlas of Protein Sequence and Structure*, 1978, ed. Dayhoff, M.O., pp. 353-358.
- [4] G.Stolovitzky and A.Califano: "Pattern Statistics in Biological Datasets," IBM RC , also subm. to "Discrete Applied Mathematics Series," editor P.Pevzner
- [5] A.Califano and G.Stolovitzky: "Pattern Discovery in Gene Expression Arrays," IBM RC
- [6] A.Califano, I.Rigoutsos: "FLASH: A Fast Look-up Algorithm for String Homology," in *Proceedings of Symposium on Intelligent Systems for Molecular Biology 1993*, Washington.
- [7] M.O. Dayhoff, R.M.Schwartz, and B.C.Orcutt: "A Model of Evolutionary Change in Proteins," *Atlas of Protein Sequence and Structure*, 1978, ed. Dayhoff, M.O., pp. 345-352.
- [8] L.Wang and T.Jiang: "On the complexity of multiple sequence alignment," *J.Comp.Biol.* 1(4):337-338, 1994
- [9] A.Brazma et al.: "Approaches to the Automatic Discovery of Patterns in Biosequences," *J.Comp.Biol.* 5(2):279-305, 1998
- [10] I.Jonassen, J.F.Collins, D.G.Higgins. "Finding flexible patterns in unaligned protein sequences," *Protein Science* 4, 1587-1595 (1995)
- [11] T.L.Bailey and M.Gribskov, "Methods and statistics for combining motif match scores," *Journal of Computational Biology*, Vol. 5, pp. 211-221, 1998.
- [12] I.Rigoutsos and A.Floratos: "Motif Discovery without Alignment or Enumeration," in S.Istrail, P.Pevzner, and M.S.Waterman editors, *Proc. 2nd annual ACM Intl. Conf. on Comp. Mol. Biol, RECOMB98*, 221-227, 1998
- [13] Altschul, S.F., Gish, W, Miller, W., Myers, E.W., and Lipman, D.J. (1990) *J.Mol.Bio.*, 215, 403-410
- [14] S.Henikoff and J.G.Henikoff: "Amino acid substitution matrices from protein blocks." *Proc. Natl. Acad. Sci. USA.* 89: 10915-10919, Nov., 1992.
- [15] Neuwald, A.F., and Green, P., "Detecting Patterns in Protein Sequences," *J. Mol. Bio.* **239**, 698-712, (1994).
- [16] Pearson, W.R. and Lipman, D.J. (1988) *Proc. Natl. Acad. Sci. USA*, **85**, 2444-2448.
- [17] M.S.Waterman, "Introduction to Computational Biology," Chapman & Hall, London, 1995
- [18] F.C.Bernstein, T.F.Koetzle, G.J.B.Williams, E.F.Meyer Jr, M.D.Brice, J.R.Rodgers, O.Kennard, T.Shimanouchi and M.Tasumi "The Protein Data Bank: A Computer-based archival file for macromolecular structures" *J. Mol Biol* (1977) **112 535-542**
- [19] Johan C.H.Gerretsen, "Lectures on Tensor Calculus and Differential Geometry," ed. P.Noorthoff N.V.Groningem, 1962
- [20] C.Cerf, G.Lippens, S.Mulyldermans, A.Segers, V.Ramakrishnan, S.J.Wodak, K.Hallenga, L.Wyns, "Homo- and heteronuclear two-dimensional NMR studies of the globular domain of histone H1: Sequential assignment and secondary structure," *Biochemistry* Vol. 32, 11345, 1993.
- [21] V.Ramakrishnan, J.T.Finch, V.Graziano, P.L.Lee, R.M.Sweet "Crystal Structure of globular domain of histone H5 and its implications for nucleosome binding," *Nature* Vol. 362, 219, 1993.