BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–10

BioSpike: Efficient search for homologous proteins by indexing patterns

Eugene Ie[a] *and Yoav Freund*[a]

[a] *Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0114, USA*

ABSTRACT

Motivation: Since the availability of high throughput sequencing tools, the number of known protein sequences has been growing at an unprecedented rate. On the other hand, information about structure or function of proteins is extremely sparse. Biologists that study proteins make extensive use of protein search engines to find homologous sequences whose structure or function are known. One well known measure for sequence similarity is the Smith-Waterman (SW) alignment score. As calculating the SW score is computationally expensive, various approximations for finding homologous sequences have been suggested, and of these the current de-facto standard for protein searching are the BLAST and PSI-BLAST methods of Altschul et al. While BLAST is an efficient approximation algorithm to the optimal SW alignment, it is still, from a computer science standpoint, a very inefficient method as it compares the query sequence to each and every sequence in the database.

Results: We present a method for indexing and searching proteins using amino acid patterns. As a source of patterns, we use the BLOCKS library of Henikoff and Henikoff [18]. Position specific scoring matrices are used to identify pattern occurrences. Each iteration consists of a "scan" in which we identify all statistically significant pattern occurrences in the sequence set; and a refinement stage, in which we use the identified occurrences to define better PSSMs. The final refined PSSMs are then used to index proteins in the UniProt Knowledgebase (UniProtKB), creating an efficient and accurate tool for searching protein homologues.

Availability: http://biospike.ucsd.edu/

Contact: yfreund@ucsd.edu

1 INTRODUCTION

At the time of writing, there are about 2,600,000 sequences in the UniProtKB data set [12, 10]. Out of these, only about 200,000 sequences in Swiss-Prot [12] have an extensive biological annotation, and only about 35,000 sequences have an associated 3D structure in the Protein Data Bank (PDB) [5]. In other words, we now know the exact amino acid (AA) sequence of many proteins, but for most of these proteins we know nothing about their structure or their function.

The Smith-Waterman (SW) score [30] is a well-known sequence similarity measure. Despite the quadratic time complexity for computing the SW alignment score, the SW score has been central to many sequence-based homology search methods. As the size of sequence data sets grow into the millions, the time complexity of SW-based scoring methods becomes a serious bottleneck. Subsequently, BLAST, PSI-BLAST, and other BLAST-like methods [2, 3, 25, 33] are developed to relieve the computational complexity of SW computation essentially through approximations.

While BLAST is an efficient approximation algorithm to the optimal SW alignment, it is extremely slow when compared to search engines such as Google which indexes billions of text documents. BLAST compares the query sequence to each and every sequence in a selected data set. In computer science, this is called exhaustive search. PSI-BLAST refines BLAST by constructing a profile after each iteration to improve search sensitivity, but the underlying mechanisms are equivalent and the computational cost is higher.

For a database with 2.6 million sequences, a PSI-BLAST search with 6 iterations (and all other default parameters) takes approximately 11.5 minutes per sequence. Using our method, an exhaustive index search takes approximately 5 seconds to retrieve an average of 70% of the sequences returned by PSI-BLAST (along with other sequences that are not detected by PSI-BLAST). With the current efforts in whole genome sequencing, the number of known protein sequences is quickly reaching a level that would make exhaustive search prohibitively slow. As far as we know, the most recent work on improving the speed of BLAST is based on clustering sequence databases into smaller sets, effectively reducing the search space required for processing a query sequence [20].

We propose an index-based search using AA patterns as indexes. These AA patterns are represented by position specific scoring matrices (PSSMs) [16]. Indexes are used in database systems as a fast way for accessing all records with a particular property. For example, so-called "inverted indexes" are used to specify all of the documents in which a particular word occurs. Using an inverted index, the user can quickly find all documents that contain all words in a given set.

The BLAST-like alignment tool (BLAT) by Kent [21] approached the problem with a similar solution by using an index of non-overlapping k-length sequence patterns (k-mers) over a species' genome before matching k-mers found on a query sequence. BLAT can perform protein homology searches 50 times faster than BLAST, but BLAT fails to compete with BLAST in remote protein homologues searches. BLAT indices are based on perfect matches of short fixed-length k-mers (default length 5). Comparing the query sequence to a remote homologue, it is quite likely to have a mutation in each k-mer, making it impossible to find the match through the index.

What we need is to replace the k-mers of BLAT with a different type of pattern, which would be less sensitive to common mutations and thus able to find the commonalities between remote homologues. We base our choice on what we learned from researchers in phylogenomics. We learned that when protein sequences are aligned, it is usually the case that there are segments of 30-80 consecutive residues that can be aligned well, interspersed with "filler" regions that do not align. Biologists commonly refer to these segments as "highly conserved regions", because the prevailing understanding is that they correspond to parts of the protein sequence that are important for its structure or function and therefore are conserved through evolution.

However, even highly conserved regions do not preserve their residues *exactly*. According to the neutral theory of molecular evolution [22], most mutations do not change fitness. This is probably true also for highly conserved regions in the following way: For most locations along the preserved sequence, there are several alternative residues that provide the required functionality. Thus even if most point mutations degrade the fitness of the protein, there can still be exponentially many variants of the conserved region that have equivalent fitness. Within this large set of variations we expect to see variations due to neutral genetic drift.

In BLAST, such variations are taken into account in a simple way. BLAST uses a single substitution matrix that captures the basic relationships between different AAs. This matrix was derived using set(s) of related sequences [7, 19]. By assuming a single substitution matrix for sequence similarity scoring, BLAST effectively ignores the effect of context. In other words, while molecular selection theory predicts that the set of equally fit substitutions for a particular location depends on neighboring residues, BLAST ignores that dependence by using a single substitution matrix for all sequence locations. Following in line with this argument, PSI-BLAST's major improvement in terms of sequence search sensitivity is greatly attributed to the profiles that are generated after each iteration of the algorithm. These profiles provides the correct context for future iterations.

There are two ways to represent highly conserved regions: as a model (a PSSM), or as a list of sequence fragments. These two representations are complementary and we use them both. In fact, our process of pattern refinement is based on alternating between the two representations. This is very reminiscent of the EM method [8], which is used extensively in statistics and machine learning. In the **scanning step** we map each PSSM to a set of sequence fragments and in the **refinement step** we map the sets of sequence fragments to a new, and hopefully better, set of PSSMs.

The scanning step is conceptually simple: for a given multiple sequence alignment, we construct a PSSM and calculate the score it gives for each location in a protein database. We then find an appropriate threshold for the PSSM so that all scores above the threshold have high statistical significance. A location in which the score is higher than the threshold is a **detection**. We store on file the location of all detections. This file forms the index which enables us to perform fast searches.

The refinement step is more involved. Here we take all the detections and estimate a new, and hopefully better, set of PSSMs. There are two parts to the refinement: estimation and clustering. Estimation is a necessary step: here we take a set of multiply aligned sequence fragments and generate a PSSM which will be a good detector for these conserved patterns. Clustering is a more elaborate step, which is not always necessary, but can lead to great improvements. The goal of clustering is to identify different patterns that are slight variants of each other and combine them into a single PSSM, thus increasing coverage.

The rest of the paper is organized as follows. We first present some basic results comparing BioSpike and PSI-BLAST protein searches. We then present our methodology for analyzing pattern significance and how it is used in the scanning and detection phase. We then show our algorithms for pattern refinement. Finally, we present the underlying components of our BioSpike search engine, showing the methods for evaluating protein queries and ranking protein homologues.

2 RELATED WORK

One of the main motivations of this work is the need to speed up BLAST or PSI-BLAST searches while maintaining a similar level of accuracy. To this end, we choose to represent conserved AA patterns using PSSMs as they are compact and also sufficient in modeling *statistically significant* short gaps.

Our method is significantly different from BLAT as we allow a more flexible pattern index in the form of varying length PSSMs. Furthermore, we also lifted the restriction for indexes to be non-overlapping. Currently with the limited set of clustered and derived motif patterns from the BLOCKS database, we are able to achieve 55% coverage of UniProtKB. By including our other derived zone patterns, we can increase our coverage to 80%. We are confident that coverage can be further improved by incorporating more structural and functional sequence motif libraries available today. This leads us to another advantage of our method — the underlying statistical

framework for analyzing any sequence alignments. Our framework allows biologists to incorporate any form of sequence patterns they may find interesting and we will use our statistical measures to determine the significance of the patterns over a large sequence database. In this way, our index is very extensible and statistically robust.

Much of the work on modeling the sequences corresponding to protein families is based on Hidden Markov models (HMMs) [28, 4, 6, 17, 23, 9]. HMMs provide the ability to model AA insertion and deletion patterns. However, computing the score of a sequence with respect to an HMM requires much more computation than calculating the score for a PSSM. We believe that, in fact, much of the flexibility of HMMs is unnecessary. Typical protein sequences consist of alternating highly conserved regions and unconserved or "filler" regions. Insertions and deletions in the conserved regions are rare and short and can be effectively ignored. On the other hand, filler regions have little statistical consistency and therefore can be treated as a concatenation of unconstrained or "wild-card" AAs. We did some exploratory analysis of multiple sequence alignments from the well-known Pfam database [11] and found the above description to be quite accurate even though the alignment was done *using* HMMs. Our conclusion is that using a much simpler statistical model which consists of free-floating PSSMs can capture the same statistical information as HMMs at a fraction of the computational cost.

In some respects, our work is very similar to the Conserved Domain Database (CDD) [27], Conserved Domain Architecture Retrieval Tool (CDART) [15], and Conserved Domain Search (CD-Search) [26] programs. These programs process query protein sequences with PSSMs using Reverse Position Specific-BLAST (RPS-BLAST) and then do various search related tasks based on the matched PSSMs.

We differ from these methods in several ways. Most importantly, perhaps, is that we use the statistics of *uncurated* protein sequences to refine our motifs. The number of uncurated sequences is much larger than the curated ones, and is likely to continue to grow rapidly in coming years. By using these much larger datasets we can generate PSSMs whose coverage and significance are much higher than those of CDD.

As we are measuring the significance of our motifs using *uncurated* sequences it seems possible that some of the PSSMs that we find have high statistical significance but no biological significance. While this is certainly possible, it seems to us very unlikely. That is because the statistical significance and the prevalence of the protein segments in the database (assuming the database has been screened for replicates) has to have an evolutionary explanation, i.e. an explanation in terms of fitness. This fitness might not be the fitness of the host organism, it might correspond to some parasitic life form such as a virus. Nevertheless, in order to be preserved as a protein code it seems necessary that the sequence segment manifests itself as part of an actual protein not too infrequently and that this manifestation does, in some way, contribute to the proliferation of the sequence segment.

In terms of a PSSM match's statistical significance, CDD, CDART and CD-Search rely on the local alignment E-value score generated by RPS-BLAST and IMPALA [29]. While IMPALA's method for evaluating statistical significance is based on estimating empirical parameters for the extreme value distribution [1, 3], our method builds the statistical model by a comparison between the null distribution of scores (similar to eMATRIX [31]) and the empirical distribution of the scores (similar to BLOCKS). In this way, our method is more general and is not restricted to any single optimal SW local alignment.

3 RESULTS

This Results Section contains two parts. In the first we demonstrate that BioSpike can generate homology sets containing most of the set created by PSI-BLAST at a fraction of the computational cost. In the second we describe the results of two specific searches demonstrating that BioSpike can identify useful remote homologies that PSI-BLAST does not detect.

3.1 Comparison with PSI-BLAST

With a small set of patterns clustered and refined from the BLOCKS database (18,134 derived BioSpike patterns), we set the minimum significance of pattern detections to 100 (see Section 4) in order to cover 55% of the UniProtKB dataset. We extract 128 random sequences with at least 250 residues from this subset. Our benchmark comparison method is PSI-BLAST (version 2.2.13) with 6 iterations and all other default parameters. Using a single CPU machine running at 2.8 GHz, PSI-BLAST processes all 128 sequences in 17 hours. With the same system configuration, BioSpike processes all 128 sequences in 10 minutes. This is approximately a 100 fold improvement in speed.

To evaluate our search results, we compare the intersection sizes between the PSI-BLAST returned sequences (P) and the sequences returned from BioSpike (B). The retrieval rate id defined as $r = \frac{|P \cap B|}{|B|}$ and is measured at different levels of cut off for B. We note that this measure assumes the PSI-BLAST result set to be the gold standard, and does not account for homologous sequences returned by BioSpike but not by PSI-BLAST.

Figure 1 shows the retrieval rate at different levels of $|B|$ cut off. Table 1 shows the results of *keywords-directed* search. Keywords-directed search is when only a subset of the patterns found on the query sequence is used to search for homologous proteins in the database. On average, we can retrieve 72% of PSI-BLAST returned sequences with an approximately 100 fold improvement in speed.

3.2 Results from specific searches

In this section we give two examples of specific searches.

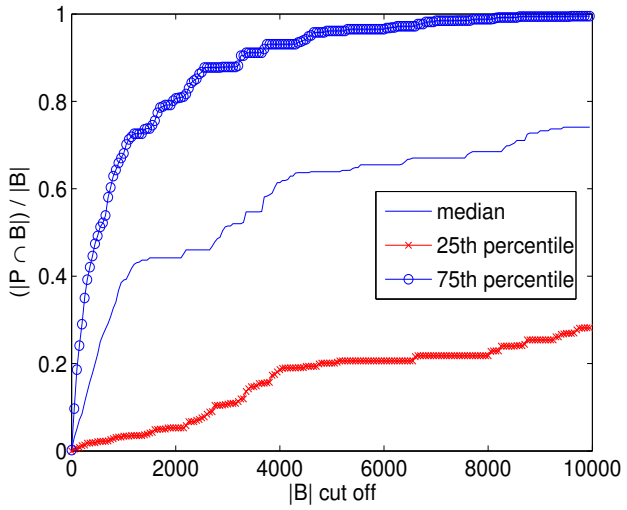

Fig. 1. The intersection size between PSI-BLAST returned sequences (P) and BioSpike returned sequences (B). The intersection is calculated at different levels of cut offs for the result set B.

Table 1. BioSpike keywords-directed search statistics. Patterns are ranked by lowest coverage first, and k is the number of pattern "keywords" to incorporate in a protein database search.

k	search time (s)	mean r	stdev r
1	3.5	0.4745	0.3527
2	3.6	0.6229	0.3360
3	3.7	0.6707	0.3390
4	3.8	0.6931	0.3462
5	3.8	0.6980	0.3475
6	3.9	0.7027	0.3472
all	5.0	0.7165	0.3479

3.2.1 Identifying the Legume lectin domains in a putative receptor-type protein kinase LRK1 We analyze the putative receptor-type protein kinase LRK1 extracted from the rice genome of *Oryza sativa*, accession number Q8H4I6. This protein has 636 amino acids and contains Legume lectin domains.

A PSI-BLAST search with 6 iterations and all other default options took 10.5 minutes and did not find any sequences with PDB annotations. With BioSpike, the sequence search took 8 seconds and found 9 PDB annotated sequences within the top 400 result sequences. Legume lectin alpha and/or beta domains are found in 8 of the 9 PDB annotated sequences.

The BioSpike template that was detected on these PDB annotated sequences is a cluster of 3 BLOCK patterns: Legume lectin alpha domain 1 (IPB000985F), Legume lectin alpha domain 2 (IPB000985G), and Legume lectin beta domain 1 (IPB001220G). BLOCK IPB000985F is the anchor pattern on the template; BLOCKs IPB000985G and IPB001220G are patterns in position 36 of the template. To illustrate the versatility of the BioSpike template, we show a multiple alignment of the PDB sequence regions indexed by this template in Figure 2. Clearly, the BioSpike template captured the various gaps between the two distinctive Legume lectin domains. Figure 3 also illustrates the 3-dimensional structures for 4 of the PDB annotated regions generated by QuickPDB.

```
CONA_CANVI_              QNGKVGTAHIIYNSVGKRLSAVVSYPNG------DSATVSYDVDLDNVLPEWVRVGLSASTGL
CONA_CANBR_              QNGKVGTAHIIYNSVGKRLSAVVSYPNG------DSATVSYDVDLDNVLPEWVRVGLSASTGL
LECA_DIOGU_              -TGKVGTAHISYNSVAKRLSAVVSYTGS------SSTTVSYDVDLNNVLPEWVRVGLSATTGL
LECA_DIOGR_              -TGKVGTVHISYNSVAKRLSAVVSYSGS------SSTTVSYDVDLNNVLPEWVRVGLSATTGL
LEC1_CRAMO_              -NGKVGTAHISYNSVAKRLSAVVSYPGG------SSATVSYDVDLNNILPEWVRVGLSASTGL
Q9ZTA9_DOLLA_            -NGKIATAHISYNSVSKRLSVTSYYAGS------KPATLSYDIELHTVLPEWVRVGLSASTGQ
Q8W1R6_GRISI_            -SGKIATARISYDGSAEILTVVLSYPDG------SDYILSHSVDMRQNLPESVRVGISASTGN
LEC_PEA_                 -NGEEANVVIAFNAATNVLTVSLTYPNSLEEENVTSYTLSDVVSLKDVVPEWVRIGF------
Q8H4I6_ORYSA(QUERY)      -GGGPIQVWIEYDGYTTLTNVTIAPLGMAR---PIRPLLSVTRDLSTVFTNQSYLGFSSS---
                           *   . * ::.    ..    .          :*   .:   ..:   :*:
```

Fig. 2. The multiple alignment of pattern regions for PDB annotated sequences returned by BioSpike. The fragment labeled q is from the query sequence.

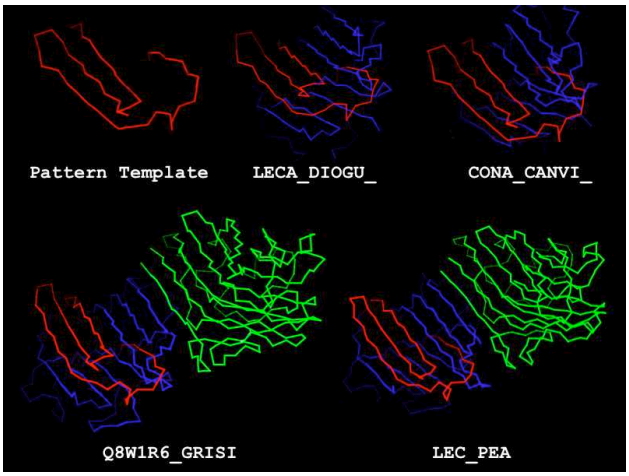

Fig. 3. Legume lectin domains in several BioSpike result sequences. The protein sequences are colored in blue, and the detected Legume lectin domains are highlighted in red. Other related PDB chains are colored in green. The upper left image is the approximate structure of the BioSpike pattern used to detect Legume lectin domains.

3.2.2 Identifying the Enoyl-CoA hydratase site of a putative enzyme We analyze the putative enzyme extracted from the bacterial genome of *Shigella flexneri*, accession number

Q83QQ0. This protein has 714 amino acids and contains a domain with a Enoyl-CoA hydratase site.

A PSI-BLAST search with 6 iterations and all other default settings took 11.5 minutes and did not find any sequences with PDB annotations. With BioSpike, the sequence search took 5 seconds and found a PDB annotated sequence with the Enoyl-CoA hydratase site. Using the PDB sequence returned, one can easily infer the structure of the Enoyl-CoA hydratase site in the putative enzyme.

4 STATISTICAL SIGNIFICANCE OF AMINO ACID PATTERNS

PSSMs are generalized probabilistic models that have been used in many successful applications as models to encode rich representation of local AA environments [24, 32]. They are derived from AA frequencies observed at specific locations over a sequence set. We describe a methodology for analyzing the statistical significance of any PSSMs.

When observing over a large sequence data set, we define a PSSM to be interesting and useful when it assigns high scores only to a small fraction of the possible AA positions (*high significance*) and it covers a sufficiently large set of sequences (*high coverage*). These two properties ensure that the PSSMs we use are non-redundant and they can be used to create a useful index to address most sequences in the data set.

We denote a PSSM as a $20 \times k$ matrix P with rows corresponding to the twenty possible AAs, and columns representing the k positions along the pattern. Given a protein sequence $x = x_1x_2...x_n$, we define the *score* for position $1 \leq i \leq n-k+1$ as $s_i = \sum_{j=0}^{k-1} P_{x_{i+j},j}$. In this work we restrict the PSSM entries to three values $\{-1, 0, +1\}$. Where $P_{ij} = 1$ corresponds to a preference *for* AA i in position j, $P_{ij} = -1$ corresponds to a preference *against* AA i and $P_{ij} = 0$ corresponds to no preference.

For a given PSSM P, we define two probability distributions over scores: the *null* distribution and the *empirical* distribution. The *null* distribution, denoted $\tilde{p}$, is the distribution of scores that results from drawing the AAs $x_1...x_k$ independently at random according to their background probabilities (i.e. the frequency of AAs on the entire dataset). As the PSSM has only three possible values $\{-1, 0, +1\}$, the null score distribution for a given PSSM can be computed in time $O(k)$ using convolutions. The *empirical* distribution $\hat{p}$ is the distribution of scores over the complete UniProtKB database. Exact calculation of this distribution requires scanning all locations in all proteins in the database. However, we can get sufficiently accurate estimates of $\hat{p}$ using a random sample containing 10% of the sequences.

We define the significance of a PSSM as the ratio between the number of locations that receive a high score to the number of such locations predicted by the null distribution. We define the significance for score s as $\theta_s = \frac{\hat{p}(\mathcal{S} \geq s)}{\tilde{p}(\mathcal{S} \geq s)}$. See Figure 4 for a typical example of the relationship between $\hat{p}(\mathcal{S} \geq s)$ and $\tilde{p}(\mathcal{S} \geq s)$.

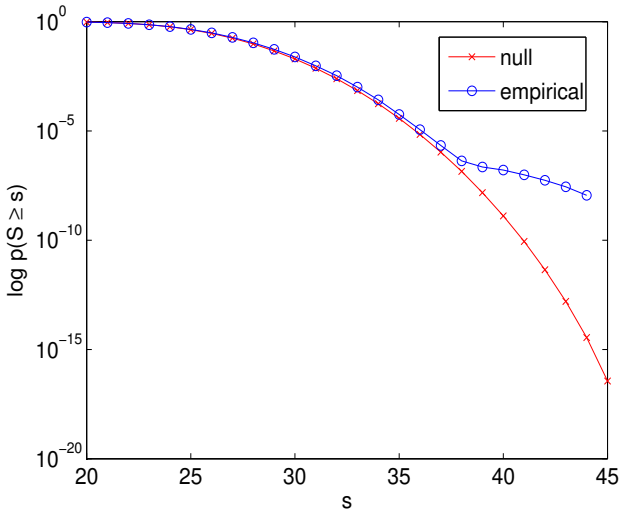

Fig. 4. The tails of the empirical and estimated theoretical score distributions for the PSSM corresponding to the the RuvA BLOCK motif in Bacterial DNA recombination protein (InterPro BLOCK Id:IPB000085A) [14]

4.1 Mapping the Sequences

After establishing statistical significance models for all patterns available, we detect occurrences of patterns by setting an arbitrary universal threshold $\theta > 1$. For pattern m, the critical score for detection is,

$$S_{m,\theta} = \arg\min_s \left[\frac{\hat{p}(\mathcal{S} \geq s)}{\tilde{p}(\mathcal{S} \geq s)} > \theta \right].$$

Regions in proteins that score above $S_{m,\theta}$ are statistically significant locations for pattern m. We will analyze these locations in Section 6 to form pattern families.

5 PATTERN REFINEMENT

Given a set of aligned segments, all corresponding to the same pattern, we produce a PSSM for detecting these segments. Our goal is to design a PSSM which assigns to the given segments scores that are much higher than the scores that are expected for the null score distribution for the same PSSM. Recall that the null distribution assumes that the residues in the pattern are independent of each other. Thus making the scores for the given set of segments much higher than expected from the null distribution implies that the PSSM has captured strong dependencies between the residues.

We start by computing the a frequency matrix which defines the empirical distributions of AA in each location of the alignment. We then translate the frequency matrix into a PSSM,

independently for each position, using a process of iterative model refinement.

Each column in the frequency matrix is modeled as a mixture of the background AA distribution and several δ-distributions. We define a δ-distribution as a probability distribution concentrated on one of the twenty possible AAs. Denote the background probability and δ-distribution of amino acid i as q_i and δ_i, respectively. We model the acid's empirical probability as,

$$\tilde{h}_i = \pi_0 q_i + \pi_i \delta_i,$$

where π_0 and π_i are mixture ratios. By modeling the frequency column as a mixture, we deduce which AAs are *essential* to a position in the pattern.

We build the mixture model iteratively, starting with a pure background distribution and adding δ-distributions in a greedy fashion until the model is sufficiently similar to the empirical distribution and that the remaining discrepancy can be attributed to sampling error. As the number of sequence fragments grows, the discrepancy that is attributable to sampling error becomes smaller, which enables us to reliably infer the essentiality of more AAs.

The KL divergence is a well known dis-similarity measure between two probability distributions, and it is defined as $D_{KL}(h||\tilde{h}) = \sum_i h_i \log(h_i/\tilde{h}_i)$. Our algorithm first approximates the distribution h using the background distribution q. Then it proceeds by iteratively adding δ-distributions to AAs that minimizes the KL divergence between our approximation and the true empirical distribution. The algorithm is terminated by a threshold inversely proportional to the number of sequence fragment windows, n. In this way, we place more confidence in refinements derived from more sequence fragments. See Figure 5 for the details of our algorithm.

For each PSSM column, we use our algorithm to compute the set of δ-distributions, Δ, and map the estimated empirical model, $\tilde{h}$, into an integer column, z, using the following rules,

$$z_i = \begin{cases} 1 & \text{if } h_i > q_i \text{ and } i \in \Delta, \\ -1 & \text{if } h_i < q_i \text{ and } i \in \Delta, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Figure 6 illustrates a sample refinement for the Guanine-specific ribonuclease N1 and T1 BLOCK (InterPro BLOCK Id: IPB000026A). The original profile contains amino acid probability distributions for each position of the BLOCK alignment. We refine the information encoded in the original profile into a matrix with values representing amino acid preferences (or non-preferences). One can argue that our discretization may discard important information about the "degree" of amino acid preferences (or non-preferences). However, we believe that this simplification leads to score distributions that are more easily interpretable.

1: Input c, n:
2: $\Delta = \{\}$
3: $i \leftarrow 0$
4: $\tilde{h}^i \leftarrow q$
5: **while** $D_{KL}(h||\tilde{h}^i) > c/n$ **do**
6: **for** each amino acid j **do**
7: $\tilde{h}^{i'} \leftarrow \tilde{h}^i$
8: $\tilde{h}^{i'}_j \leftarrow h_j$
9: $\Delta' \leftarrow \Delta \cup \{j\}$
10: Re-normalize non-Δ' elements in $\tilde{h}^{i'}$
11: $d_j \leftarrow D_{KL}(h||\tilde{h}^{i'}) - D_{KL}(h||\tilde{h}^i)$
12: **end for**
13: $k \leftarrow \arg\max_j d_j$
14: $\tilde{h}^{i+1} \leftarrow \tilde{h}^i$
15: $\tilde{h}^{i+1}_k \leftarrow h_k$
16: $\Delta \leftarrow \Delta \cup \{k\}$
17: $i \leftarrow i + 1$
18: **end while**
19: Return Δ, $\tilde{h}^i$

Fig. 5. Algorithm for computing the estimated empirical model for a PSSM column and the list of AA with δ-distributions in the model. The parameter c is tuned so that the average number of δ-distributions per column is approximately 5.

6 GROUPING SEQUENCE PATTERNS

The Haemagluttinin motif (InterPro Id: IPR008635) and the Hep_Hag motif (InterPro Id: IPR008640) are observed to co-occur on bacterial haemagglutinins and invasins with high probability. We now show a framework for finding groups of biological patterns that co-occur and co-locate with high probability.

Using the pattern detections over the UniProtKB data set, we perform pattern co-occurrence and co-location analysis to define rough *templates* that capture sets of highly specific patterns. Our goal is to discover significant rough patterns with high coverage. These rough patterns will allow us to partition the search space, enabling fast and directed sequence searches.

6.1 Motifs and Zones

We analyze pattern similarity by looking at the locations of pattern detection. Before computing the pattern similarity matrices for clustering, we need to partition the set of patterns into two classes, which we call *zones* and *motifs*.

Zones are patterns which are usually detected at several consecutive positions along a sequence. For example, the polyglutamine tracts found in proteins of *Dictyostelium discoideum* are examples of zones [13]. On the other hand, motifs are highly specific patterns which are usually detected at isolated well defined locations. Motifs are the more interesting patterns as they define a highly specific conserved region which are likely to correspond to a specific 3D structure in the protein. In contrast, zones define regions in which there

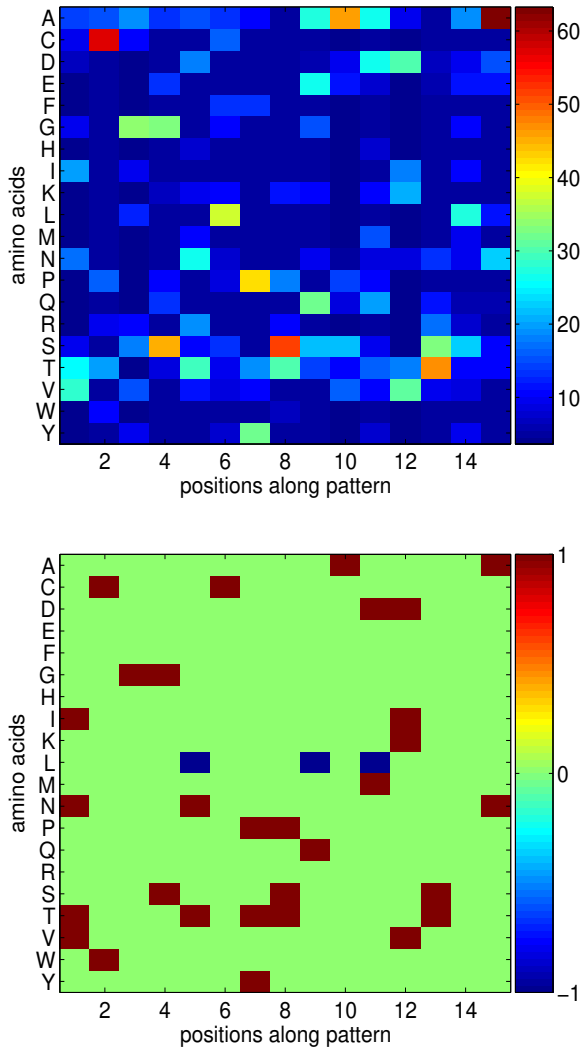

Fig. 6. PSSM estimation for IPB000026A BLOCK. The original profile (**top**) and the computed PSSM (**bottom**).

is an abundance of some types of amino acids. While zones have clear statistical significance, their biological significance is unclear.

We can partition the set of patterns into the two classes by analyzing the set of detections for each pattern. A motif is characterized by having a clear peak score location on the protein sequences, whereas a zone will have multiple peak score locations situated in close proximity.

6.2 Similarity Matrices for Sequence Motifs

The **co-occurrence matrix** O describes how likely a pattern m_i is detected with another pattern m_j on a sequence x. The elements of the co-occurrence matrix O are defined as

$$o_{i,j} = \frac{\log\left(P(m_i, m_j)/(P(m_i)P(m_j))\right)}{\log\left(1/\max\{P(m_i), P(m_j)\}\right)}, \qquad (2)$$

where $P(m_i)$ is the fraction of sequences that contain an occurrence of the pattern m_i and $P(m_i, m_j)$ is the fraction of sequences that contain occurrences of both m_i and m_j. The range of $o_{i,j}$ is $(-\infty, 1]$, where $o_{i,j} = 0$ if m_i and m_j are independent, and $o_{i,j} = 1$ if one of the patterns *dominates* the other, i.e. pattern m_i always occur when pattern m_j occurs, or vice versa. Getting $o_{i,j} < 0$ means that the m_i and m_j are *anti-correlated*. In this work we ignore information about anti-correlations.

Similarly to O, we compute the **co-location matrix** L using the same formula but redefining $P(m_i)$ to be the fraction of sequence *locations* in which m_i is detected and $P(m_i, m_j)$ to be the fraction of detections of m_i (m_j) such that a detection of m_j (m_i) occurs in a close proximity, i.e. the two patterns either overlap or are within a distance of at most k from each other. In our experiments, we set $k = 10$.

6.3 Clustering Motifs

We cluster the set of motifs in two steps. Clusters found using O are re-clustered using matrix L. In this way, the clustering algorithm at the second stage exclusively analyzes co-occurring motifs. Any clustering algorithm can be applied on matrices O and L. We use a simple clustering scheme which first removes entries in the matrices O and L based on some positive thresholds θ_O and θ_L. Then, we interpret each of these matrices as a graph connectivity matrix with nodes representing sequence motifs and edges representing the strength of the relation (co-occurrence or co-location score). By computing the reachability of each node on either graphs, we can group protein motifs into clusters. To prevent having weak links form large clusters of loosely connected sequence motifs, we also require that each motif cluster satisfy a certain *tightness* threshold. This tightness threshold is simply the average value over the cluster's sub-matrix in O or L.

6.4 Motif Template Construction

We partition clusters into three types: *singleton*, *fixed gapped*, and *variably gapped*. Singleton clusters are motifs that have no observed relationships with any other motifs. As the name suggests, fixed gapped clusters contain motifs that co-locate at a specific gap from each other. For variably gapped clusters, motifs are observed to co-locate at varying gaps, possibly as a result of residue insertions/deletions or as a result of cyclical patterns.

PSSMs in singleton clusters are refined using the new counts but their length is maintained. PSSMs in gapped clusters are combined into "templates" which are then used to estimate new and longer PSSMs.

6.4.1 Analyzing Common Gaps Between Protein Motifs. We define a *template* to be the minimal consensus window that can be used to arrange motifs from a cluster in frequently observed relative positions.

Denote a cluster of motifs as $\mathcal{C}$. A template is the set of (motif, location)-pairs $\mathcal{T} = \{(m_i, l_{i,j})\}$ where $m_i \in \mathcal{C}$, and

$l_{i,j} \geq 0$ is the jth location of pattern m_i in the template. Note that $|\mathcal{T}| \geq |\mathcal{C}|$ for variable gapped clusters because a motif can occur several times in the template. If $\mathcal{C}$ is a fixed gapped cluster, then $|\mathcal{T}| = |\mathcal{C}|$.

A histogram of gaps is computed for all pairs of motifs in the cluster. Denote a gap as g, and its associated count as c. For template construction, we aggregate the histograms into a single set forming, $\mathcal{H} = \{(m_a, m_b, g, c) \mid m_a \in \mathcal{C}, m_b \in \mathcal{C}\}$.

We use a greedy algorithm to construct the template. The algorithm iteratively incorporates pairwise motif relationships onto the template with the most frequently observed relationships first. The algorithm stops when the fraction of unsatisfied constraints drops below a specified threshold. Adding a single relationship to the template can result in adding several new (motif, location)-pairs if the template has variable gaps, as one of the motifs in the relationship might already occur in several locations in the template.

6.4.2 Constructing Generalized PSSMs for Pattern Families. After constructing a template $\mathcal{T}$, we fit sequence fragments from motif detection sites onto the template consensus window. Since a sequence x can have several fragments that fit onto the template window, a fractional weight is assigned to each fragment to avoid over-representation of a single sequence. The multiply aligned sequence fragments are then used to generate a PSSM for the motif family using methods described in Section 5.

Finally, we scan the template PSSMs over the entire sequence data set for another round of pattern detection. Figure 7 shows a comparison in significance of detections and UniProtKB database coverage between the original BLOCKS derived profiles and the refined template profiles. Coverage of the UniProtKB data set is increased, the significance of detections is also increased, and we reduced the number of patterns needed to index the data set. We can run additional refinement steps. However, the current version of the BioSpike index is based on a single refinement.

6.5 Zone Templates

As zones are detected at consecutive locations along a sequence, it is not possible to analyze co-location or co-occurrence of zoning patterns using the methods described for motifs. Instead of marking each location of zone detections individually, we define a center and radius for zone detections on protein sequences.

After mapping the centers and radii for each zone's detections on sequences, we construct an AA probability distribution h by observing the AA frequencies on zone windows. The AA distribution for a zone will be very different from the background probability distribution of AAs over all sequences. For example, the probability of glutamine in zones that capture polyglutamine tracts will be much higher than the background probability of glutamine.

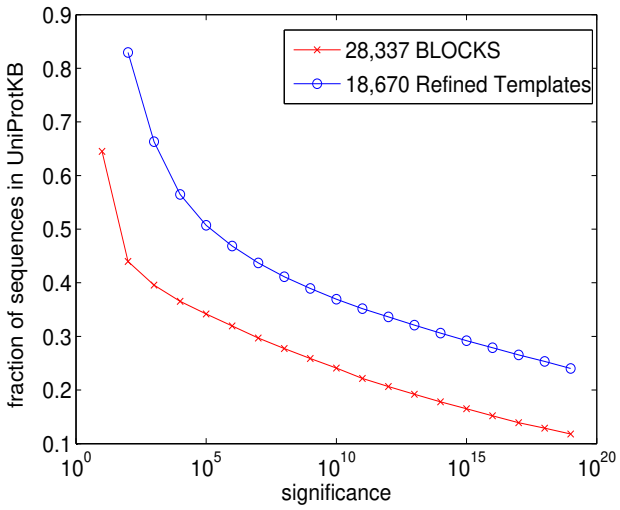

Fig. 7. A comparison of significance of detections and UniProtKB protein coverage between BLOCKS derived profiles and refined template profiles.

After constructing an AA distribution for each zone, we use the same algorithm described in Figure 5 to translate it into integer columns, z. We then cluster the set of z, grouping them based on agreements on the z_i elements. Zones that have a disagreement in preference/non-preference of a particular amino acid is separated. Finally, we construct a template for each cluster of zones by computing another integer column, z', using the aggregated AA probability distribution for zones in the cluster. Then by repeating columns z' over the average length of zone PSSMs in the cluster, we define a generalized zone template PSSM.

7 PROTEIN SEARCH ENGINE

7.1 Building BioSpike

In general, the search engine development iterates over PSSM refinement, sequence database scan, pattern detection, and pattern clustering, in which the pattern detection stage builds the database index. For our proof-of-concept prototype, we use the 28,337 ungapped multiple alignments from BLOCKS database (version 14.1) as seed patterns, and run two iterations to generate 18,670 motif templates and 34 zone templates.

To organize an inverted index of motif and zone templates to UniProtKB (release 6.5) sequences, we use the open source *MySQL* relational database. The motif and zone templates are kept in two separate lists, where they are sorted by protein coverage. Each template has an associated critical detection score as defined by the last sequence database scan.

7.2 Query Processing

Query processing operates in three phases: *pattern matching*, *database search* and *results ranking*.

In the pattern matching phase, we scan all patterns indexed by BioSpike over the query sequence. This phase generates a list of detected patterns, which are sorted ascendingly according to UniProtKB coverage sizes.

There are many variations of database search one can do given a set of detected patterns. For example, the keywords-directed search as described in the Results Section limits the number of patterns in the database query. Since the patterns are sorted by ascending database coverage, as more "keywords" are included into the database query search, more remote homologues will be included in the result set. For our online search engine (http://biospike.ucsd.edu/), we include all patterns in the search, allowing the user to select a result set size cutoff and bringing all detections that are in PDB to the top of the list.

For results ranking, we use a simple ranking function which compares the query sequence and result sequence in two perspectives: pattern detection score differences and pattern order differences.

Define the list of detected patterns in the query as an ordered list $\mathcal{A} = [(p_0, s_0), \ldots, (p_n, s_n)]$, and the list of detected patterns on a result sequence as an ordered list $\mathcal{B} = [(\hat{p}_0, \hat{s}_0), \ldots, (\hat{p}_m, \hat{s}_m)]$. For each list, we retain a single entry for each unique pattern p_i (or $\hat{p}_j$) with the maximal score. We denote these reduced lists as $\mathcal{A}'$ and $\mathcal{B}'$.

To capture pattern detection score differences, we define $\mathcal{I} = (\mathcal{A}' \cap \mathcal{B}')_{|p}$ as the intersection between the lists induced by $p_i = \hat{p}_j$. Finally, we define σ and τ as ordered lists of patterns in $\mathcal{A}'$ and $\mathcal{B}'$, respectively, padding the shorter of the two with the missing patterns.

Using our notations, the ranking function is defined as,

$$rank(\mathcal{B}') = \frac{1}{|\mathcal{I}|} \sum_{p_i = \hat{p}_j} |s_i - \hat{s}_j| + C'_n(\sigma, \tau),$$

where $C'_n(\sigma, \tau)$ is the normalized Spearman footrule distance between the permutations σ and τ. Intuitively, the first term in the ranking function accounts for the difference in pattern detection scores, whereas the second term accounts for the difference in pattern arrangements in the primary sequences.

8 DISCUSSION

We have presented a novel method for searching protein homologues using amino acid patterns as indexes. Our method is approximately 100 times faster than PSI-BLAST whilst maintaining a similar level of accuracy. Moreover, the ability of BioSpike to identify remote homologues is superior to that of PSI-BLAST. The main reason that BioSpike is so fast is that we rely heavily on pre-computation. By pre-computing and storing on disk the locations of pattern detections, we remove the need to perform similar computations repeatedly. Pre-computation also creates a dramatic reduction in the time it takes the system to respond to a new query. This reduction in query processing time can, we believe, make the system much more attractive as an interactive exploration tool for the research biologist.

We are considering many directions for continuing this work. This list includes enlarging, refining, and curating the set of motifs; creating data structures to speed up the search for motifs in a new query sequence; using motifs as features for classifying proteins; and identifying which motifs have consistent 3D structure and can thus be seen as "building blocks" of proteins.

However, on the short term, our most important goal is to make the BioSpike search engine as user-friendly as possible so that, instead of trying to pursue all of these research directions ourselves, we can let others, more knowledgeable in protein structure than us, use the tool to advance their research.

ACKNOWLEDGEMENTS

We thank the FWGrid project for providing the computational resources necessary for this work. We also thank Burkhard Rost and Kimmen Sjölander for many helpful discussions on this work. Thanks to Adam Godzik for bringing CDS related work to our attention. The molecular images are produced using the QuickPDB program provided by the PDB group.

REFERENCES

[1]S. F. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

[2]S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[3]S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[4]P. Baldi, Y. Chauvin, T. Hunkapillar, and M. McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the United States of America*, 91:1059–1063, 1994.

[5]H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[6]M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. *Proceedings of First International Conference on Intelligent Systems for Molecular Biology*, pages 47–55, 1993.

[7]M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. *Atlas of Protein Sequence and Structure*, volume 5, chapter A model of evolutionary change in proteins: matrices for detecting distant relationships, pages 345–358. National Biomedical Research Foundation, 1978.

[8]A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, 39:1–38, 1977.

[9]S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.

[10]A. Bairoch *et al.* The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 33:D154–D159, 2005.

[11]A. Bateman *et al.* The Pfam Protein Families Database. *Nucleic Acids Research*, 32:D138–D141, 2004.

[12]B. Boeckmann *et al.* The Swiss-Prot Protein Knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31:365–370, 2003.

[13]L. Eichinger *et al.* The genome of the social amoeba *Dictyostelium discoideum*. *Nature*, 435:43–57, 2005.

[14]N.J. Mulder *et al.* The InterPro Database. *Nucleic Acids Research*, 31:315–318, 2003.

[15]L. Y. Geer, M. Domrachev, D. Lipman, and S. H. Bryant. CDART: protein homology by domain architecture. *Genome Research*, 12(10):1619–1623, 2002.

[16]M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 84(13):4355–4358, 1987.

[17]D. Haussler, A. Krogh, I. S. mian, and K. Sjölander. Protein modeling using hidden Markov models: Analysis of globins. *Proceedings of the Hawaii International Conference on System Sciences*, 1:792–802, 1993.

[18]S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19:6565–6572, 1991.

[19]S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89:10915–10919, 1992.

[20]M. Itoh, S. Goto, T. Akutsu, and M. Kanehisa. Fast and accurate database homology search using upper bound of local alignment scores. *Bioinformatics*, 21(7):912–921, 2005.

[21]W. J. Kent. BLAT: The BLAST-like alignment tool. *Genome Research*, 12(8):656–664, 2002.

[22]M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217(129):624–626, 1968.

[23]A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[24]R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of Bioinformatics and Computational Biology*, 3(3):527–550, 2005.

[25]B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[26]A. Marchler-Bauer and S. H. Bryant. CD-Search: protein domain annotations on the fly. *Nucleic Acids Research*, 32:W327–W331, 2004.

[27]A. Marchler-Bauer, A. R. Panchenko, B. A. Shoemaker, P. A. Thiessen, L. Y. Geer, and S. H. Bryant. CDD: a database of conserved domain alignments with links to domain three-dimensional structure. *Nucleic Acids Research*, 30(1):281–283, 2002.

[28]L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. pages 267–296, 1990.

[29]A. A. Schaffer, Y. I. Wolf, C. P. Ponting, E. V. Koonin, L. Aravind, and S. F. Altschul. IMPALA: matching a protein sequence against a collection of PSI-BLAST-constructed position-specific score matrices. *Bioinformatics*, 15(12):1000–1011, 1999.

[30]T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[31]T. D. Wu, C. G. Nevill-Manning, and D. L. Brutlag. Fast probabilistic analysis of sequence function using scoring matrices. *Bioinformatics*, 16:233–244, 2000.

[32]G. Yona and M. Levitt. Within the twilight zone: A sensitive profile-profile comparison tool based on information theory. *Journal of Molecular Biology*, 315(5):1257–1275, 2002.

[33]Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational Biology*, 7(1–2):203–214, 2000.