[Frontiers in Bioscience 14, 5052-5065, June 1, 2009]

Computational techniques for motif search

Sanguthevar Rajasekaran

Dept. of CSE, University of Connecticut, Storrs, CT 06269-2155

TABLE OF CONTENTS

1. ABSTRACT

In the study of protein-protein interactions, one is interested in identifying domains as well as short motifs that bind to these domains. Short motifs also confer functions to proteins such as post-translational modification, protein-protein interaction, and protein trafficking. Identification of domains is relatively easy since they are sufficiently long enough to render the likelihood of occurrences by random chance very low. On the other hand, the identification of motifs is a challenge since they are typically very short. Thus it is vital to develop efficient techniques to identify motifs. In this paper we survey some of the techniques that have been proposed in the literature for motifs identification.

2. INTRODUCTION

Motif search is an important problem in biology since it has numerous applications including the study of protein-protein interactions. Numerous approaches have

been proposed in the literature for solving this crucial problem. These techniques can perhaps be categorized into two, namely, experiments-based and computing-based. In this paper we provide a summary of some of the computing-based techniques that have been employed in motif search. We also briefly summarize experiments-based techniques.

Genomic sequencing allows the identification of nearly all of the genes in an organism, from which the potential gene products can be inferred. However, genomic sequencing does not provide direct information on the function of the individual gene products, nor their interrelationships. Current approaches for predicting the function of proteins provide important clues, but we need more approaches. Homology analysis of protein sequences has proven effective in inferring protein function, most notably by facilitating the identification of similar protein domains in different genes and organisms (see e.g., PROSITE and SMART) (1-2). The function of the domain can then be inferred from previously characterized proteins and subsequently confirmed in the uncharacterized proteins. However, this approach requires that a query has homology to a domain that has already been characterized. As protein domains are highly conserved throughout evolution, it is logical to expect that their binding partners or substrates would be conserved as well. The domain may contain short functional sites (such as protein interaction sites) that can indicate the function. These functional sites are what we refer to as motifs in this paper. These motifs are difficult to find just based on conservation alone since short conserved segments can occur in multiple species by random chance. However, as a first step in the process of identifying motifs, we can obtain a list of closely conserved short sequences that occur in multiple species. There could be many false positives in this list. We could then apply various filters to trim this list down further. This general theme is used by many scientists.

In this paper we focus on motif search algorithms. We can categorize motif search techniques into two: experimental and computational. Experimental techniques, by nature, are very time consuming. However, whether a putative motif is real or not can only be confirmed with experiments. Even if we have an upper bound on the length of any motif, since there could be a large number of possible strings, experimentally verifying if each of these strings is a real motif or not is impractical. This is where the computational techniques could help significantly. Computational techniques can process through all possible strings of a given length and identify a small number of them as putative motifs which then can be experimentally evaluated. Though a major goal of this paper is to discuss computational techniques for motif search, we provide a brief introduction to experimental techniques in Section 3. Section 4 is devoted to computational techniques.

3. EXPERIMENTAL TECHNIQUES

When computational techniques identify putative motifs, we can confirm these with biochemical experiments. Though experimental techniques are very accurate, they are very costly and time consuming. Computational techniques can be used to reduce the cost and time significantly. Functional subdomains can be identified experimentally using cross-linking, assay of proteolytic fragments, the study of mutants, etc. (3). Motifs can be found from crystal structure data. Experimental techniques have been used to find binding sites as well. Experimental techniques include analyzing protein complexes, phage display, and mutagenesis (4-6).

Edwards, *et al.* address the problem of discovering novel bioactive peptides (7). Short synthetic oligopeptides that encompass functional motifs play an important role in the study of protein signaling and interactions. The authors specifically targeted oligopeptides that could identify human platelet function. Toward this goal they have employed high-throughput *in vitro* platelet function assays and have been able to identify many agonists and antagonists of platelet function.

4. COMPUTATIONAL TECHNIQUES

In this Section we discuss computational techniques that have been proposed for motif search. Computational techniques can be classified into two, namely, discrete algorithmic methods and stochastic (or statistics based) methods. In general, stochastic methods are non-optimal or non-complete algorithms. These algorithms perform well in practice. On the other hand, discrete algorithmic techniques typically focus on providing optimal and/or complete results. Run times of some of these techniques could be long. For example, the run times could be exponential in some of the underlying parameters. In Section 4.1 we introduce statistics based methods and Section 4.2 is devoted for discrete algorithmic techniques.

4.1. Statistics based techniques

Several statistical techniques have been used to identify putative motifs. For example, Austin, *et al.* have applied statistical techniques to identify tripeptide motifs in the C-termini (8). They point out that C-termini are often sites of activity for a variety of biologically important functions and hence the authors concentrate on identifying motifs in C-termini. Their basic approach is to estimate the over-representation of position-specific tripeptides in seven eukaryotic proteomes using z-statistic. In order to reduce background noise they employ randomization models and masking. Before estimating the over-representation they cluster the proteins into gene-families and the analysis is performed within individual families. This is done to eliminate errors that might be caused by counting irrelevant representations of any motif. Comparative genomics has then been applied to identify tripeptides that are over-represented across many species. They report a good success with their approach. For example, they have been able to predict all the C-terminal targeting motifs present in the literature (at the time of publication). Deng, *et al.* have used maximum-likelihood method to estimate domain-domain interactions (9). Reiss and Schwikowski build a simple probabilistic model based on protein sequence

information and all versus all protein interaction datasets to identify ligand peptides of peptide recognition modules (10).

The algorithms PRINTS, BLOCKS, GibbsSampler, and MEME also fall under the statistics based methods. None of these techniques is optimal but give a reasonably good set of candidate results with high statistical significance. For example, MEME works as follows (11). Given a set of input sequences, it fits a two-component finite mixture model to these sequences using expectation maximization. This algorithm also takes as input the length of the motif under concern. It is capable of finding multiple motifs. MEME also outputs a model and a threshold for each motif. This model and the threshold can be used to find occurrences of the same motif in other databases.

4.2. Discrete algorithmic methods

4.2.1. Web-based systems

Several web-based systems are available for motif search. In this section we provide details on some of them.

4.2.1.1. ELM

ELM is a system developed for investigating candidate short non-globular functional motifs in eukaryotic proteins (12). A user can input any protein sequence and the species information. The user, optionally, can also submit relevant sub-cellular compartment(s). The ELM server then identifies plausible motifs. The main algorithm used by ELM is one of sequence comparisons. It looks for substrings in the query sequence that are very similar to the already known motifs. These matching substrings serve as candidates for output. However, sequence similarity alone might result in too many false positives. Thus ELM uses a series of filters to eliminate as many false positives as possible. Even then it may not eliminate all of the false positives. In fact computational techniques alone cannot eliminate all the false positives. Experimental confirmation is the ultimate proof of a motif. False positives could arise because the sequence matches occur in an irrelevant context or because they match to a wrong cellular compartment (12).

ELM keeps a database with tens of tables that store information relevant for linear motifs. All data input is handled with hand curation. Annotation of a motif involves literature search, similarity search using BLAST, multiple sequence alignment of relevant sequences, etc. Motif patterns are represented as POSIX regular expressions. Two annotation standards, namely, Gene Ontology (GO) and the NCBI taxonomy database identifiers are used.

Some of the filters used by ELM are: 1) Cell Compartment Filter: Each motif in ELM is annotated with cell compartments in which the motif is known to function. If the user inputs the compartments in which the query sequence is known to function, ELM will restrict its output to only matches in these compartments; 2) Globular Domain Filter: Matches inside globular domains are not reported. SMART and Pfam databases are used for this filtering; and 3) Taxonomic Filtering: Each motif in ELM is annotated with NCBI taxonomy node identifiers to denote its known phylogenic relations. Motifs that are not related to the lineage of the species input by the user are filtered out.

4.2.1.2. SCANSITE

Yaffe, *et al.* focus on the problem of inferring protein function and predicting protein-protein interactions (13). They point out that identifying signaling domains within protein sequences is a simpler problem than identifying short linear motifs targeted by the domains. They propose a technique for the discovery of motifs based on peptide-library based search.

When a user inputs a query sequence, SCANSITE scans through the sequence looking for potential motifs. A sliding window of size 15 is used for this purpose. A score is computed for each substring of length 15. Within any substring, residues critically invariant for a motif are identified. A position-specific score is assigned for each residue in a window and a normalized score is obtained for each putative motif. This score, S_{raw}, is then compared with the best possible score, S_{opt}, for each motif giving a final score of $S_f = (S_{opt} - S_{raw}) / S_{opt}$. The S_f value of each motif site within the query sequence is then compared to pre-calculated scores of the motif for all possible sites and for all vertebrate sequences in SWISS-PROT to obtain a rank for the motif site. This site will be considered a hit if its rank exceeds a threshold value.

For any given query sequence, SCANSITE provides a lot of information in the output. For example, it displays domain information (obtained using Pfam). Motif sites are displayed with a vertical bar the length of which is proportional to the corresponding score. An abbreviated name of the motif is also shown. A plot of the surface accessibility is also a part of the output. If one clicks on the protein cartoon, a table that carries detailed information on the motif sites, their scores, ranks, etc. is displayed.

4.2.1.3. PROSITE

Hulo, *et al.* have developed the web system PROSITE that stores protein domains, families and functional sites (14). It also stores the associated patterns and profiles that can be used to identify them. PROSITE has defined a set of rules, called ProRules, based on patterns that provide valuable information about functionality. A ProRule is nothing but a set of conditional annotations in Swiss-Prot format. The condition could be on the position-specific presence of amino acids, the presence of other domains, etc. The version of PROSITE dated November 11, 2007 has 1319 patterns, 745 profiles, and 764 ProRules.

For any given query sequence, PROSITE reports matches to the patterns it has in its database. In earlier versions of PROSITE, the matches were reported without

any indication of the significance of the matches. The latest version uses a new method to estimate the significance of matches.

4.2.1.4. MnM (Minimotif Miner)

Balla, *et al.* have constructed a database of short motifs and their web system MnM (http://sms.engr.uconn.edu) helps users in the identification of motifs in proteins (15). They employ three different scores to score a candidate motif, namely, frequency score (FS), surface prediction score (SPS) and evolutionary conservation score (ECS). The web module of the MnM application has a database with data integrated from the current versions of RefSeq, LocusLink, HomoloGene and Taxonomy databases of NCBI. The protein sequences in RefSeq are screened for all minimotifs in the MnM database using a motif search algorithm for proteome of different species, namely, human, mouse, rat, yeast, rice, fruitfly, malaria parasite and watercress. Pre-processed minimotif statistics are stored in the MnM application database as follows. The total occurrences of each amino acid are measured for each proteome independently, the probability of finding a given amino acid X in a proteome being given by ($\sum AA_X / \sum AA_{TOT}$) where AA_X is the total count of the amino acid X in the proteome and AA_{TOT} is the total count of all the amino acids in the proteome. The expected count (C_e) of a minimotif in the proteome is given by the product of the probabilities of the individual or subsets of amino acids defining the minimotif and the possible number of occurrences of the minimotif in the proteome dataset. Let (C_a) be the actual count (i.e.) the actual number of occurrences of the minimotif in the proteome dataset. The enrichment factor *efactor* of a minimotif is defined as the ratio of its actual count to its expected count in the proteome dataset (*efactor*= C_a / C_e). For minimotifs occurring in the N- and the C- termini, the positional probabilities of the amino acids is considered while calculating the expected count of the minimotif. Information stored in the MnM application database includes statistical details of the distribution of minimotifs present in the proteomes of various species, the actual count, the expected count and the enrichment factor of each minimotif.

User-entered accession numbers retrieve protein details and minimotif information from the MnM application database. The user has the choice of entering three different types of accessions, namely, RefSeq Accession, SwissProt Accession or GI of a protein. Also, search criteria include the species for which minimotif statistics are to be retrieved and the sub-cellular localization of the minimotifs. For entries pasted into the sequence input window, the motif search algorithm searches the query sequence for each minimotif, recording its count and positions in the protein sequence. Sequences are sorted based on the motif score and output to the minimotif table. Three different scores can be used. For example, a 'Frequency Score' *FS* of a minimotif in a protein sequence of length *x* is arrived as follows: $ms = C_a(p) / C_e(p)$, where $C_a(p)$ is the actual number of occurrences of the minimotif in *p* and $C_e(p)$ is the expected number of occurrences of the minimotif in *p*. The denominator is $C_e(p) = C_e * (x/m)$, where *m* is the total number of amino acid residues in the proteome dataset.

The output screen of the search displays a table of protein details and provides links to various NCBI search sites. Minimotifs appear highlighted on the protein sequence at positions they occur. A table of minimotif statistics that ranks the minimotifs in descending order of the motif scores is also available. This table contains the actual count, the expected count and the motif score of the minimotif in the protein sequence, the actual count, the expected count and the enrichment factor of the minimotif in the proteome of the species for which the query was submitted, the positions of occurrence of the minimotifs in the protein and links to references available online for the minimotif.

The user also has choices of viewing the domains (the "View Domains" button) and the SNPs (the "View SNPs" button) present in the protein. A table of minimotifs similar to the one explained above can be retrieved for the set of protein sequences in the group of the query protein in the HomoloGene database by clicking on the "View Homologenes" button. Individual minimotif can be highlighted on the protein sequence by choosing the radio button that belongs to it and using the "View Selected Motif" option. Also, there is a pop-up that displays information of minimotifs, domains and SNPs when the user moves the mouse over the highlighted portions of the protein sequence.

Version 2 of MnM has been released in November 2008 (http://mnm.engr.uconn.edu).

4.2.1.5. Other web systems

There are other web systems such as DiLiMoT (Discovery of Linear Motifs) and SMART (Simple Modular Architecture Research Tool) (16, 2). 3MOTIF is a web application that can be used to visually map motifs onto 3D protein structures in the PDB (17). Crucial properties of the motifs are displayed with different colors in this map.

4.2.1.6. A comparison of different web systems

Table 1 provides a comparison of various web based systems for motif search. ELM uses sequence analysis as a primary technique and hence it can be categorized under the discrete algorithmics technique. The filters it uses exploit biological information available in various databases such as GO. The query engine of MnM uses sequence analysis. This part of MnM uses discrete algorithmic techniques. For filtering it employs some statistics based techniques such as frequency analysis. Similar to ELM, MnM also employs some known biological databases to remove false positives. SCANSITE computes a score for each putative motif and it can be thought of as a purely statistics based method. On a given query sequence, PROSITE uses string matching algorithms to report matches to the patterns it has in its database. Thus PROSITE is also based on discrete algorithmic techniques.

4.2.2. Algorithmic techniques

The algorithmic aspects of motif search have been addressed by numerous researchers. Depending on the

data used by these techniques they can be classified into two, namely, those that employ *interaction data* and those that use *singular sequence data.*

4.2.2.1. Interaction data algorithms

Interaction data can be used to discover binding sites. Binding sites might give us clues on the motifs present. Many research works concentrate on solving this problem. For example, Li and Li focus on finding the two sides of binding (18). The two sides are referred to as a *binding pair.* The authors concentrate on finding pairs where each side has a short sequence of continuous residues. Such pairs are called *short sequence motifs.* They first find *maximal contact segment pairs* from a protein complex structural dataset. These segment pairs are then used to cluster the structural dataset and as a result form candidate short sequence motifs. Finally, an iterative refinement is applied on the candidate motifs to finalize the motifs.

Nedua, *et al.* concentrate on the study of protein interaction networks (16). The authors point out that not all protein-protein interactions are mediated by pairs of globular domains. Many such interactions happen over the binding of the domain of one protein with a short sequence (referred to as a *linear motif*) in the other. These authors also point out that linear motifs are difficult to find since they are short and tend to reside in disordered regions in proteins. Their method for detecting linear motifs is based on the hypothesis that the linear motifs are detectable by the virtue of over-representation. For a given set of proteins that share an interaction partner, they first preprocess the sequences to eliminate regions (such as globular domains, coiled-coils, collagen regions, etc.) that cannot possibly contain the linear motifs. They then generate all three to eight-mers in the remaining sequence. The overrepresentation of each such substring is calculated in comparison with a random sequence. The authors claim that their technique results in the rediscovery of known motifs and the prediction of new ones.

Tan, *et al.* discover correlated linear motifs from sparse and noisy protein interaction data (19).

4.2.2.2. Singular sequence data algorithms

Algorithms that make use of singular sequence data exclusively can further be classified into those that look for arbitrary patterns and those that look for specific patterns. In this paper we concentrate on three specific patterns, namely, *Simple Motif Search (SMS), (l,d)-motif search (LDMS) and Edit-distance based motif search (EMS).* Section 4.2.2.3 considers arbitrary patterns algorithms. Sections 4.2.2.4, 4.2.2.5, and 4.2.2.6 are devoted for SMS, LDMS, and EMS, respectively. In Section 4.2.2.7 we give a brief introduction to algorithms that find *spaced motifs,* i.e., motifs that have several segments separated by non-conserved characters.

4.2.2.3. Algorithms for finding arbitrary patterns

Brazma, *et al.* is an excellent survey paper that discusses the problem of identifying arbitrary patterns automatically from biological (DNA, RNA and protein) sequences (20). Patterns of interest are Generalized Regular Patterns. An example of a PROSITE pattern is C-x (2,4)-C-x (3)- (LIVMFYWC)-x (8)-H. This stands for any string that starts with C, followed by 2 to 4 arbitrary symbols, followed by C, followed by 3 arbitrary symbols, followed by a member of {L,I,V,M,F,Y,W,C}, followed by 8 arbitrary symbols, followed by H. Many different algorithmic approaches have been employed by researchers to identify arbitrary patterns. We enumerate some of them next.

4.2.2.3.1. Learning

Brazma, *et al.* pose the problem of finding patterns as a learning task (20). Two problems are considered. The first problem is to find a classifier function for a family of sequences. This function will take as input a query sequence and output YES if the sequence is a member of the family and NO otherwise. The corresponding learner will be trained with both positive and negative examples. The function to be learnt should have a "short description" and it should be correct with high probability. The second problem takes as input a collection of sequences belonging to the same family and the goal is to infer patterns characteristic of the family. These authors also acknowledge that the ultimate test of whether a pattern is biologically significant or not lies only in experimental verifications. This paper provides a summary of prior works that address variants of the pattern identification problem.

4.2.2.3.2 Bottom-up processing

Jonassen, *et al.* present a bottom-up approach for identifying patterns (21). Any bottom-up approach enumerates all possible patterns, calculates the "fitness" of each pattern, and finally outputs the best scoring patterns. There are a number of ways for defining the *fitness of a pattern*. Fitness of a pattern, for example, could be the number of example sequences in which the pattern either occurs exactly or approximately. Jonassen, *et al.* represent the pattern space as a tree and perform a depth-first search of this tree (21). They also employ an elegant data structure called *block data structure* that enables one to efficiently find the set of substrings matching each pattern. This algorithm is able to identify patterns having both ambiguous positions and gaps. An improved version of this algorithm has been given by Jonassen (22). For example this improved version enables the use of branch-and-bound and heuristics to make the search efficient.

Jonassen & Eidhammer and Jonassen report variations of the above algorithm called *PRATT* and *PRATT2* (23, 22). These algorithms define a *pattern graph.* Each path in this graph corresponds to a set of patterns. Here also, a depth-first search through the graph is made looking for *conserved patterns* that have the highest fitness scores. A pattern is said to be conserved if it occurs in at least N_{min} of the positive examples given. Here N_{min} is a user-specified number.

4.2.2.3.3. Top-down processing

Top-down approaches for finding patterns are based on optimally aligning the input sequences and

enumerating longest common subsequences. Since the problem of multiple sequence alignment is known to be NP-complete, several heuristics are employed for aligning sequences. The algorithm of Smith and Smith uses the dynamic programming algorithm to construct a dendrogram for the sequences (24). This dendrogram can be thought of as an estimated phylogenic tree. Each leaf of this dendrogram corresponds to an input sequence. The dendrogram is a binary tree where at each internal node a pairwise alignment is performed to identify the common pattern between the two children of this node. A child could either be a pattern or an input sequence. Pairwise alignments are always performed with sequences or patterns that are the most similar. For example, if the input sequences are S_1, S_2, S_3, S_4, S_5 and if S_1 is similar to S_2; S_3 is similar to S_4; and S_5 is the most dissimilar from the others, then, S_1 and S_2 will be aligned to identify the pattern P_1 between them; S_3 and S_4 are aligned to get the pattern P_2; P_1 and P_2 are aligned to get P_3; Finally P_3 and S_5 are merged to get P_4; P_4 is the pattern identified to be common among the five input sequences. If there are *k* sequences and the average length of each sequence is *n*, then the run time of this algorithm is $O(kn^2)$. Since the construction of dendrograms is a common data clustering technique, Smith and Smith's algorithm can also be used to cluster the input sequences (24).

4.2.2.3.4. Clustering approaches

Zhong, *et al.* employ clustering techniques to explore local protein sequence motifs representing common structural property (25). The authors argue that systems such as PROSITE, PRINTS, and BLOCKS are based on searching sequences for conserved patterns (including experimentally validated motifs). In other words, these systems make use of apriori information on the datasets to be analyzed and frequent human intervention. In contrast, clustering techniques do not require any such prior knowledge. Zhong, *et al.* make use of k-means clustering to cluster data so that each cluster will have proteins with similar characteristics (25). K-means clustering has been used before by Han and Baker to find recurring local sequence motifs in proteins (26). Zhong, *et al.* modify the k-means clustering algorithm with a new greedy initialization technique (25). The clustering applied by the authors is based on recurring sequence motifs. They also study the structural similarity of proteins in each cluster thus obtained.

4.2.2.3.5. Heuristics

Other heuristics are reported by Roytberg, Schuler, *et al.*, Vingron, *et al.*, etc. (28-30). Brazma *et al.* have developed a fitness measure based on minimum description length principle and use it to automatically find significant patterns in unaligned sequences (30). Here also the authors state: "Evaluating whether these relations have biological significance is outside the scope of the method and have to be explored individually in each case." More patterns finding algorithms can be found in (31-32).

In addition to the above techniques for identifying patterns, clustering, etc., three formal versions of the motif search problem have been identified in the literature see e.g., (33). In the next three subsections we introduce these problems and provide a summary of algorithms that have been employed to solve them.

4.2.2.4. Simple motif search (SMS)

A pattern is a string of symbols (also called residues) and ?'s. A "?" refers to a wild card character. A pattern cannot begin or end with ?. AB?D, EB??DS?R, etc. are examples of patterns. The length of a pattern is the number of characters in it (including the wildcard characters). The problem of SMS is to take as input a database *DB* of sequences and to identify all the patterns of length at most *P* (with anywhere from 0 to $\lfloor P/2 \rfloor$ wild card characters). All the patterns together with a count of how many times each pattern occurs should be output (34). Optionally a threshold value for the number of occurrences could be supplied.

Rajasekaran, *et al.* have derived the above motif model and this model has been derived as follows (34). They have generated a list of 312 minimotifs (i.e., motifs of short length) that have defined biological functions. They have used this list to select parameters for a de novo analysis of novel minimotifs in the human proteome. The authors suggest a value of 10 for *P*. The average minimotif in their list has 2.1 wildcard positions for any amino acid. *Wildcards signify* any of the 20 amino acids. Since only 13 % of minimotifs in the list have more than 50% wild card positions, they have chosen *P*/2 or 5 wild cards as the maximal number in the algorithm.

4.2.2.4.1. Biological significance of SMS

A problem similar to SMS has been addressed by Rigoutsos and Floratos (35). Their algorithm is called TEIRESIAS. Core histones play a central role in the packaging of DNA within the cell. Crystallographic techniques have established the presence of the core histone motif common to all the histone proteins (36). There are two families of histone proteins, namely, H3 and H4. Members of each family are known to be highly similar. However, similarity across the two families was not known until the application of TEIRESIAS. TEIRESIAS was able to identify a large number of patterns shared by the two families (35, 37). SMS algorithm has been employed in MnM to identify potential motifs which are then scored using three metrics. Davey, *et al.* have employed TEIRESIAS and BLAST to identify putative motifs (38).

4.2.2.4.2. An algorithm for SMS

Rajasekaran, *et al.* have given a simple algorithm for this problem that takes less time than that of TEIRESIAS (34). Note that SMS identifies **all** the patterns of length at most *P (*with anywhere from 0 to $\lfloor P/2 \rfloor$ wild card characters). For every pattern, the number of occurrences should be output.

Let a (*u, v*)-class be a class of patterns such that each pattern has length *u* and has exactly *v* wild card characters. For example, GA??C?T belongs to (7, 3)-class. Clearly, there are $\binom{u-2}{v}|\Sigma|^{u-v}$ patterns in a (*u*, *v*)-class. *To identify the patterns in a (u, v)-class, they perform* $\binom{u-2}{v}$ *sorts*. In particular, for each possible placement of *v* wild card characters (excluding at the end positions) in a sequence of length *u*, they perform a sorting. For example, when *u*=6 and *v*=2, there are six possible placements: C??CCC, CC??CC, CCC??C, C?C?CC, CC?C?C, and C?CC?C. Here C corresponds to any residue. Every such placement is called a (*u*, *v*)-pattern type.

For every (*u*, *v*)-pattern type, the following steps are performed.

1. If *R* is a pattern type in (*u, v*)-class, generate all possible *u*-mers in all the sequences of *DB*. If the sequences in *DB* have lengths $l_1, l_2, \ldots, l_n$, respectively, then the number of *u*-mers from S_i is $l_i - u + 1$, for $1 \le i \le n$.
2. Sort all the *u*-mers generated in step 1 only with respect to the non-wild card positions of *R*. For example, if the pattern type under concern is CC??C?C, generate all possible 7-mers in *DB* and sort the 7-mers with respect to positions 1,2,5, and 7. Employ radix sort. See e.g., (39).
3. Scan through the sorted list and count the number of occurrences of each pattern.

The run time of the above algorithm is $O\left(\binom{u-2}{v} M \frac{u}{w}\right)$ for a (*u*, *v*)-class, where *M* is the total number of residues in DB and *w* is the word length of the computer. Thus the run time of the entire algorithm is $O(P^{P/2}M)$ (35).

4.2.2.4.3. Algorithm TEIRESIAS

The TEIRESIAS algorithm addresses a problem similar to SMS (35, 37). They define a *pattern* to be a string of symbols (also called *residues*) and ?'s. A "?" is a wild card character. A pattern cannot begin or end with ?. AB?D, EB??DS?R, etc. are examples of patterns. The *length* of a pattern is the number of characters in it including the wildcard characters. For any pattern *P*, any substring of *P* that itself is a pattern is called a *subpattern* of *P*. AB is a subpattern of AB?D, for example. *P* is called a <*l, W*> pattern if every subpattern of *P* of length *W* or more contains at least *l* residues. A pattern *P'* is said to be *more specific* than a pattern *P* if *P'* can be obtained from *P* by changing one or more wild card characters of *P* into residues and/or by adding one or more residues or wild card characters to the left or right of *P*. ABCD and E?AB?D are more specific than AB?D. We call a pattern *P* *maximal* if there is no pattern *P'* that is more specific than *P* and which occurs the same number of times in a given database *DB* as *P*. The problem TEIRESIAS addresses takes as input a database *DB* of sequences, the parameters *l, W*, and *q* and outputs all <*l, W*> maximal patterns in *DB* that occur in at least *q* distinct sequences of *DB*.

The run time of TEIRESIAS is $O(W^l N \log N)$, where *N* is the size of the database (i.e., the number of characters (or residues) in the database). TEIRESIAS algorithm consists of two phases. In the first phase elementary <*l, W*> patterns are identified. An elementary <*l, W*> pattern is nothing but a pattern which is of length *W* and which has exactly *l* residues. This phase runs in time $O(NW^l)$. In the second phase (known as the convolution phase), elementary patterns are combined (i.e., convolved) to obtain larger patterns. As an example, AS?TF and TFDE can be combined to obtain AS?TFDE. All the convolved patterns that pass the support level and which are maximal will be output. The run time of this phase is $O(W^l N \log N)$.

4.2.2.4.4. Apriori principle based algorithms

Ye, *et al.* consider an extension of TEIRESIAS (40). They concentrate on three types of patterns: Type I, Type II, and Type III. Type I patterns are the same as the simple patterns defined in (35, 37). A type II pattern is a PROSOITE-like pattern. For instance, Sx(2,3)AF is a type II pattern that has three residues and a variable wildcard region. A type III pattern is of a given length and in which a set of residues occurs in a predefined order. For example, a type III pattern of length 8 could have the residues A, T, and F in this order. This pattern could be denoted as A*T*F. Six different algorithms for finding these three types of patterns are given in (40). These algorithms employ the *Apriori principle* and is similar to the pattern growth algorithm PRATT of (21). The apriori principle states that any super-pattern of an infrequent pattern cannot be frequent. This principle has been employed, for example, to identify association rules in databases (41).

The input to the algorithms are a dataset *S.* Input also are integers *min_support, min_non_wc,* and *max_wc_l.* Here, *min_support* refers to the minimum number of sequences in which the pattern should occur; *min_non_wc* is the minimum number of non-wildcard characters in each pattern; and *max_wc_l* is the maximum number of consecutive wildcard characters allowed in any pattern. A pattern is said to be *frequent* if it occurs in at least *min_support* input sequences. If a pattern is frequent and obeys the other two constraints, call it a *valid pattern.* The algorithms output all the valid patterns. The algorithm starts by finding all the valid patterns of length one each. For every such one-pattern it tries to expand it on the right by one more symbol and checks if the resultant pattern is valid or not. As a result, valid patterns of length 2 are obtained. In a similar manner, patterns of length more than 2 are also identified. Ye, *et al.* have compared their algorithms with PRATT2 and TEIRESIAS (40). Their algorithms perform better than PRATT2 in terms of

efficiency as well as the diversity of patterns found (21-22). Also, they are comparable to TEIRESIAS in terms of performance but are better in terms of the diversity of patterns.

4.2.2.5. (*l, d*) motif search (LDMS)

The second version of interest is called (l,d) motif search (LDMS) and has been introduced in (42). The input consists of *n* sequences of length *m* each. Two integers *l* and *d* are also input. The problem is to find a motif (i.e., a sequence) *M* of length *l.* It is given that each input sequence contains a variant of *M*. The variants of interest are sequences that are at a Hamming distance of *d* from *M*.

4.2.2.5.1. Biological significance of LDMS

Buhler and Tompa have employed LDMS algorithms to find known transcriptional regulatory elements upstream of several eukaryotic genes (43). In particular, they have used orthologous sequences from different organisms upstream of four types of gene: preproinsulin, dihydrofolate reductase (DHFR), metallothioneins, and c-fos. These sequences are known to contain binding sites for specific transcription factors. The authors point out the differences between experimental data and synthetic data that LDMS algorithms are typically tested with. For example, the background DNA in experimental data is not random. Their algorithm successfully identified the transcription factor binding sites. They have used the values of *l*=20 and *d*=2. The same sites have also been found using PMS2 of Rajasekaran, *et al.* (44). Buhler and Tompa have also employed their algorithm to solve the ribosome binding site problem for various prokaryotes (43). This problem is even more challenging since here the number of input sequences could be in the thousands.

Eskin and Pevzner used LDMS algorithms to find composite regulatory patterns (45). They point out that traditional pattern finding techniques (on unaligned DNA sequences) concentrate on identifying high-scoring monads. A regulatory pattern could indeed be a combination of multiple and possibly weak monads. They employ MITRA (a LDMS algorithm) to locate regulatory patterns of this kind. The algorithm is demonstrated to perform well on both synthetic and experimental data sets. For example, they have employed the upstream regions involved in purine metabolism from three Pyrococcus genomes. They have also tested their algorithm on four sets of S.cerevisiae genes which are regulated by two transcription factors such that the transcription factor binding sites occur near each other.

Price, *et al.* have employed their PatternBranching LDMS technique on a sample containing CRP binding sites in E.coli, upstream regions of many organisms of the eukaryotic genes: preproinsulin, DHFR, metallothionein, & c-fos, and a sample of promoter regions from yeast (46). They report finding numerous motifs in these sequences.

Though the (l,d) motif problem is defined for arbitrary sequences, all the algorithms in the literature assume DNA sequences. The literature contains numerous algorithms for solving the LDMS problem. These algorithms can be classified into two. Those algorithms that may not output the correct answer(s) always are referred to as *approximation algorithms (or heuristic algorithms)* and those that always output the correct answers are called *exact algorithms.*

4.2.2.5.2. Approximation algorithms

Examples of approximation (or heuristic) algorithms include Random Projection, PatternBranching, MULTIPROFILER, CONSENSUS, and ProfileBranching (43, 46, 42, 47, 46). These algorithms have been experimentally demonstrated to perform well. We provide summaries of Random Projection and PatternBranching next.

4.2.2.5.2.1. Random projection algorithm

The algorithm of Buhler and Tompa is based on random projections (43). Let the motif *M* of interest be an *l*-mer and *C* be the collection of all the *l*-mers from all the *n* input sequences. The algorithm projects these *l*-mers along *k* randomly chosen positions (for some appropriate value of *k*). We can think of the projection of each *l*-mer as an integer. We group the projected values (which are *k*-mers) according to their integer values. In other words, we hash all the *l*-mers using the *k*-mer of any *l*-mer as its hash value.

The main idea of the algorithm is the observation that many instances of *M* will have the same projected value. If a hashed group has at least a threshold number *s* of *l*-mers in it, then there is a good chance that *M* will have its *k*-mer equal to the *k*-mer of this group. Since there are $n(m-l+1)$ *l*-mers in the input and there are 4^k possible *k*-mers, the expected number of *l*-mers that have the same hash value is $\frac{n(m-l+1)}{4^k}$. The threshold *s* is chosen to have twice this expected value and *k* is chosen such that $n(m-l+1) < 4^k$. This will ensure that the expected number of random *l*-mers that have the same hash value is less than one. Also *k* has to be less than (*l-d)*. Typical values used for *k* and *s* are 7 and 3, respectively.

The above process of random hashing is repeated *r* times (for some relevant value of *r*) so as to be sure that a bucket of size $\geq s$ is observed at least once. Calculation of *r* can be done as follows. The probability *p* that a given instance of *M* has the same hash value as *M* is given by $\frac{\binom{l-d}{k}}{\binom{l}{k}}$. Since there are *n* instances of *M,* the probability that fewer than *s* of them have the same hash

value is given by $p' = \sum_{i=1}^{s-1} \binom{n}{i} p^i (1-p)^{n-i}$. Therefore, the probability that less than *s* instances have the same hash value in each of the *r* trials is $P = (p')^r$. The value of *r* is thus $\left\lceil \frac{\log P}{\log p'} \right\rceil$. Buhler and Tompa employ a value of 0.05 for *P* (43). The algorithm collects all the *k*-mers (and the corresponding *l*-mers) that pass the threshold and these are processed further to arrive at the final answer *M*. Expectation maximization (EM) technique of Lawrence and Reilly is employed to process these *l*-mers (48). The EM formulation employs the following model. Each input sequence has an instance of *M (*where $|M|=l$) such that these instances are characterized by a $4 \times l$ weight matrix *W* with $W[i, j]$ being the probability that base i occurs in position *j* for $1 \le i \le 4$ and $1 \le j \le l$. Occurrences of bases in different positions are assumed independent. Bases for the remaining *m-l* positions in each sequence are governed by a background distribution *B*. If *S* is the set of input sequences, then the EM-based technique of Lawrence and Reilly determines a weight matrix model *W** that maximizes the likelihood ratio $\frac{\Pr(S/W^*, B)}{\Pr(S/B)}$.

4.2.2.5.2.2. Pattern branching

A local searching algorithm called *PatternBranching* has been given in (46). This algorithm falls under the approximation category. If *u* is any *l*-mer, then there are $\binom{l}{d} 3^d$ *l*-mers that are at a Hamming distance of *d* from *u.* Refer to each such *l*-mer a *neighbor* of *u*. One strategy to solve the LDMS problem is to start from each *l*-mer *u* in the input, search the neighbors of *u*, score them appropriately and output the best scoring neighbor. Note that there are a total of $n(m-l+1)$ *l*-mers in the input.

Let $S = S_1, S_2, \ldots, S_n$ be the collection of *n* given input sequences. PatternBranching only examines a selected subset of neighbors of any *l*-mer *u* of the input and hence is more efficient. For any *l*-mer *u,* let $D_i(u)$ stand for the set of neighbors of *u* that are at a Hamming distance of i from *u (*for $1 \le i \le d$). For any input sequence S_j, let $d(u, S_j)$ denote the minimum Hamming distance between *u* and any *l*-mer of S_j (for $1 \le j \le n$). Let $d(u, S) = \sum_{j=1}^{n} d(u, S_j)$. For any *l*-mer *u* in the input let *BestNeighbor*(*u*) stand for the neighbor *v* in $D_1(u)$ whose distance $d(v, S)$ is minimum from among all the elements of $D_1(u)$. PatternBranching starts from a *u*, identifies $u_1 = BestNeighbor(u)$; Then it identifies $u_2 = BestNeighbor(u_1)$; and so on. It finally computes u_d. The best u_d from among all possible *u*'s is output.

4.2.2.5.3. Exact algorithms

Many exact algorithms are known as well. Examples include the ones given by Martinez, Brazma,, Galas, *et al.,* Sinha & Tompa, Staden, Tompa, van Helden, *et al.*, Rajasekaran, *et al.*, Davila & Rajasekaran, and Davila, *et al.* (49-55, 44, 56-57). However, as pointed out by Buhler and Tompa, these algorithms "become impractical for the sizes involved in the challenge problem" (43). A *challenging instance of LDMS* is an instance where the probability of finding a supurious motif (i.e., a motif that occurs by random chance) is greater than or equal to 1. Exceptions are the MITRA algorithm and the algorithms of Rajasekaran, *et al.* (45, 44, 56-57). MITRA solves for example the (15, 4) instance in 5 minutes using 100 MB of memory (45). This algorithm is based on the WINNOWER algorithm and uses pairwise similarity information (58). A new pruning technique enables MITRA to be more efficient than WINNOWER. MITRA uses a mismatch tree data structure and splits the space of all possible patterns into disjoint subspaces that start with a given prefix. The same (15,4) instance is solved in 3.5 minutes by PMS2 (44).

It is noteworthy here that approximation algorithms such as CONSENSUS and ProfileBranching take much less time for the (15, 4) instance (46). However these algorithms fall under the approximate category and may not always output the correct answer.

The largest challenging instances that have been solved thus far are (17,6) and (19,7). These instances have been solved by an algorithm called PMSprune (59). PMSprune is an exact algorithm and it can be thought of as an extension of PMS1. This algorithm takes 69 minutes and 9.2 hours to solve the instances (17,6) and (19,7), respectively.

We now provide summaries of some of the algorithms that have been proposed for LDMS.

4.2.2.5.3.1. Algorithms WINNOWER and SP-STAR

WINNOWER algorithm of Pevzner and Sze works as follows (58). If *A* and *B* are two instances (i.e., occurences) of the motif in two different sequences, then the Hamming distance between *A* and *B* is at most 2*d*. It can be shown that the expected Hamming distance between *A* and *B* is $2d - \frac{4d^2}{3l}$. WINNOWER constructs a collection *C* of all possible *l*-mers in the input. A graph $G(V, E)$ is constructed in which each *l*-mer of *C* will be

a node. Two nodes *u* and *v* in *G* are connected by an edge if and only if the Hamming distance between *u* and *v* is at most 2*d* and they come from two different sequences.

Since the *n* instances of the motif *M* will form a clique in *G*, the problem of finding *M* reduces to that of finding large cliques in *G*. Unfortunately, the problem of finding large cliques in a graph is *NP*-hard and also there will be numerous 'spurious' edges (i.e., edges that do not connect instances of *M*) in *G*. Pevzner and Sze observe that the graph *G* constructed above is 'almost random' and is multipartite and use this observation to eliminate edges (58). If $Q = \{v_1, v_2, \ldots, v_k\}$ is any clique, node *u* is called a *neighbor* of *Q* if $\{v_1, v_2, \ldots, v_k, u\}$ is also a clique. If *u* is a neighbor of *Q*, then *Q* can be extended to get a larger clique. A clique is said to be *extendable* if it has at least one neighbor in every part of the multipartite graph *G*. WINNOWER is based on the observation that every edge in a maximal *n*-clique belongs to at least $\binom{n-2}{k-2}$ extendable cliques of size *k*.

WINNOWER iteratively constructs cliques of larger and larger sizes. If *N=mn,* then the run time of the algorithm is $O(N^{2d+1})$. This algorithm runs in a reasonable amount of time in practice especially for small values of *d.* Pevzner and Sze have also given another algorithm called SP-STAR that is faster than WINNOWER and uses less memory (58). WINNOWER algorithm treats all the edges of *G* equally without distinguishing between edges based on similarities. SP-STAR scores the *l*-mers of *C* as well as the edges of *G* appropriately and hence eliminates more edges than WINNOWER per iteration.

4.2.2.5.3.2. Algorithm PMS1

This algorithm is based on radix sorting and has the following steps: 1) Generate all possible *l*-mers from each of the *n* input sequences. Denote by C_i the collection of *l*-mers from S_i, for $1 \le i \le n$; 2) For all $1 \le i \le n$ and for all $u \in C_i$ generate all *l*-mers *v* such that *u* and *v* are at a Hamming distance of *d*. Let the collection of *l*-mers corresponding to C_i be C_i', for $1 \le i \le n$; 3) Employ radix sort to sort all the *l*-mers in every $C_i', 1 \le i \le n$ and eliminate duplicates in every C_i'. Let L_i be the resultant sorted list corresponding to C_i'; and 4) Merge all the L_i s $(1 \le i \le n)$ and output the generated (in step 2) *l*-mer that occurs in all the L_i s (44). Many fundamentally new ideas that can be used to improve the performance of PMS1 have also been given in (44).

The exact algorithms of Eskin & Pevzner and Rajasekaran, *et al.* are able to solve the challenging instances (9, 2), (11, 3), and (13, 4) in a reasonable amount of time using a PC (45, 44). However for the (15,5) instance they either take a long time or call for too much of memory. Chin and Leung have proposed a technique called voting which can be thought of as a combination of the techniques of Buhler & Tompa and Rajasekaran, *et al.* (60, 43-44). They have reported solving the (15, 5) instance in around 22 minutes. In a recent work Davila, *et al.* have improved the performance of the algorithms of Rajasekaran, *et al.* with some crucial ideas (44, 57). The largest challenging instances that have been solved thus far are (17,6) and (19,7). See (56, 59). The algorithms used to solve these instances are based on PMS1 (44). PMS1 takes an approach much different from the others in the literature. It has yielded the best performance thus far. Also, PMS1 and related algorithms use simple data structures such as arrays.

4.2.2.6. Edited motif search (EMS)

The input for this problem is a database DB of sequences $S_1, S_2, \ldots, S_n$. Input also are integers *P*, *D*, and *q*. The output should be all the patterns in *DB* such that each pattern is of length *P* and it occurs in at least *q* of the n sequences. A pattern *U* is considered an occurrence of another pattern *V* as long as the edit distance between *U* and *V* is at most *D*.

4.2.2.6.1. Biological significance of EMS

EMS has applications in finding the DNA binding sites. Rocke and Tompa have used Gibbs sampling techniques to solve EMS (61). They have tested their technique on the noncoding regions of the full H.influenzae genome and found many interesting motifs. Also, since EMS is closely related to LDMS (LDMS being a special case of EMS), all the applications relevant for LDMS can also be handled by EMS techniques.

Note that EMS is more general than LDMS. From a biological point of view EMS is perhaps more relevant than EMS since in the process of evolution, inserts and deletes are common and LDMS rules out these.

4.2.2.6.2. Algorithms for EMS

An algorithm for EMS has been given by Sagot that has a run time of $O(n^2 m l^d |\Sigma|^d)$ where *m* is the average length of the sequences in *DB* and Σ is the alphabet from which the input sequences are generated (62). It is based on suffix trees and uses $O(n^2 m / w)$ space where *w* is the word length of the computer. An algorithm with an expected run time of $O(nm + d(nm)^{1+pow(\varepsilon)} \log nm)$ where $\varepsilon = d/l$ and $pow(\varepsilon)$ is an increasing concave function has been given in (63). The value of $pow(\varepsilon)$ is roughly 0.9 for protein and DNA sequences. This algorithm is also suffix-tree based.

A sorting based algorithm similar to PMS1 has been given in (34). This algorithm runs in time

Table 1. A comparison of different web-based motif search systems

System	Freq. Analysis	SNP Comparisons	# of motifs in proteomes	Links to Literature	Domain mappings	Subcellular localization
MnM	Yes	Yes	Yes	Yes	Yes	Yes
ELM	No	No	No	Yes	Yes	Yes
Scansite	No	No	No	No	No	No
Prosite	No	No	No	Yes	Yes	No
DILIMOT	No	No	No	No	No	No

$O(n^2 ml^d |\Sigma|^d)$. The space used is $O(nml^d |\Sigma|^d)$. The space used can be reduced to $O(nmd + l^d |\Sigma|^d)$. Since this algorithm uses arrays only, the underlying constants will be small and hence can be expected to perform well in practice.

4.2.2.7. Finding spaced motifs

Some classes of motifs, called *spaced motifs,* have the following property: Each motif consists of several segments located closer to each other. All the algorithms discussed above assume *monad (*i.e., single segment) motifs. A number of algorithms have been proposed in the literature for finding spaced motifs. Examples include MITRA and SPACE (45, 64). The problem of finding spaced motifs is made difficult by the presence of non-conserved characters (called *spacers*) in between two segments (i.e., monads). Several different techniques have been used to find spaced motifs. For example, Favorov, *et al.* assume that the spacers in the same motif are of the same length (65). The algorithm of Sinha and Tompa tries out all possible spacer lengths (52). The algorithm of Carvalho, *et al.* uses suffix trees to locate regularly spaced monads and combine them to form spaced motifs (66). Eskin and Pevzner have defined a special data structure called *the mismatch tree data structure* and used this in their algorithm MITRA (45). MITRA first identifies monads and then combines them to get spaced motifs.

The algorithms of Favorov, *et al.*, Sinha & Tompa, Carvalho, *et al.*, and Eskin & Pevzner have been used to find *dyads,* i.e., motifs consisting of two monads (65, 52, 66, 45). On the other hand, the SPACE algorithm has been employed to find spaced motifs with more than two monads as well (64). SPACE is similar to TEIRESIAS and it poses the spaced motif finding problem as a frequent itemset mining problem. There are three steps in SPACE. In step 1, candidate motifs are found; in step 2 these candidates are refined into spaced motifs; and in step 3, the spaced motifs are ranked using a scoring scheme and a sorted list of the spaced motifs is output.

5. CONCLUSIONS

In this paper we have provided a survey of various techniques that are currently being employed for the discovery of short motifs. Finding motifs is an important problem and there is still room for improvement as far as the computational techniques are concerned. Numerous techniques have been proposed in the literature for motif search. Each of these techniques addresses a specific aspect of motif search. We can categorize motif search methods into two: experimental and computational

Experimental techniques are very time consuming. Given the large number of putative motifs, experimental techniques may not be feasible for an exhaustive search. This is where computational techniques could be of great help.

Computational techniques could be categorized into two: stochastic and discrete algorithms based. Stochastic techniques (STs) are often fast but may not find complete or optimal results. On the other hand, discrete algorithmic techniques (DATs) typically find optimal solutions. Some of the problem formulations in DATs are NP-hard and hence can take a very long time to solve in practice.

Based on the data used to find motifs, DATs can further be classified into those that employ interaction data and those that make use of singular sequence data. Algorithms that use interaction data are typically very effective in finding domains whereas those that employ singular sequence data are effective in finding short motifs.

DATs that use singular sequence data can further be categorized into those that find arbitrary patterns and those that look for specific patterns. Algorithms that find arbitrary patterns make use of techniques such as learning, bottom-up processing, top-down processing, clustering, and heuristics.

Three specific patterns of interest are simple motifs and variants (such as type I, type II, and type III patterns), (l,d) motifs, and edit-distance based motifs. We can think of simple motifs as a slightly restricted form of arbitrary patterns. Patterns here are represented using PROSITE-like expressions. (l,d) motifs primarily assume substitutions whereas edit-distance based motifs permit substitutions, inserts, and deletes. Algorithms for all these three specific patterns have been used to solve some important problems in biology.

Algorithms known for (l,d) motif search (LDMS) can be grouped into two: approximate and exact. An exact algorithm always comes up with the correct answer (s). An approximate (or heuristic) algorithm may not always come up with the correct answer(s). Algorithms such as Random Projection, PatternBranching, MultiProfiler, Consensus, Weeder, ProfileBranching, etc. fall under the approximation group. Examples of exact algorithms include WINNOWER, SP-STAR, MITRA, PMS1, PMS2, PMS3, Voting, PAMPA, PMSprune, etc. Approximation algorithms, in general, are very fast but may not always come up with the correct answer(s). On the other hand, exact algorithms perform an exhaustive search and hence always produce correct answer(s). By nature,

they take a long time to terminate. In fact the largest challenging instances of LDMS that have been solved so far are by an exact algorithm (PMSprune).

Some of the motif techniques assume DNA sequences. Examples include algorithms for LDMS and EMS. These algorithms will work on protein sequences as well. However, the run times of some of these algorithms are exponential in the size of the alphabet. Therefore, they will take a very long time on protein sequences. All the algorithms that have been described in Section 3.2.1 (web based systems) work on protein sequences.

In conclusion, computational techniques are of great help in motif search. The particular choice of technique will depend on various factors such as the type of data used, performance measures of interest, types of patterns desired, etc.

6. ACKNOWLEDGEMENTS

This research has been supported in part by the NIH Grant 1R01GM079689-01A1 and NSF Grant ITR-0326155.

7. REFERENCES

1. Falquet, L, M. Pagni, P. Bucher, N. Hulo, C. J. Sigrist, K. Hofmann, A. Bairoch: The PROSITE database, its status in 2002. *Nucleic Acids Res* 30, 235-238 (2002)

2. Letunic, I, L. Goodstadt, N. J. Dickens, T. Doerks, J. Schultz, R. Mott, F. Ciccarelli, R. R. Copley, C. P. Ponting, P. Bork: Recent improvements to the SMART domain-based sequence annotation resource. *Nucleic Acids Res* 30, 242-244 (2002)

3. Hodgman, T. C: The elucidation of protein function by sequence motif analysis, *CABIOS Rev* 5 (1), 1013 (1989).

4. Josephson, K, N. J. Logsdon, M. R. Walter: Crystal structure of the IL-10/IL-10R1 complex reveals a shared receptor binding site. *Immunity* 15, 35-46 (2001)

5. Sidhu, S.S, W. J. Fairbrother, K. Deshayes: Exploring protein-protein interactions with phage display. *Chembiochem* 4, 14-25 (2003)

6. Clemmons, D. R: Use of mutagenesis to probe IGF-binding protein structure/function relationships. *Endocrine Reviews* 22, 800-817 (2001)

7. Edwards, R. J, N. Moran, M. Devocelle, A. Kiernan, G. Meade, W. Signac, M. Foy, S. D. E. Park, E. Dunne, D. Kenny, D. C. Shields: Bioinformatic discovery of novel bioactive peptides. *Nature Chemical Biology* 3 (2), 108-112 (2007)

8. Austin, R. S, N. J. Provart, S. R. Cutler: C-terminal motif prediction in eukaryotic proteomes using comparative genomics and statistical over-representation across protein families. *BMC Genomics*, 8:191 (2007)

9. Deng, M, S. Mehta, F. Sun, T. Chen: Inferring domain-domain interactions from protein-protein interactions. *Genome Res* 12, 1540-1548 (2002)

10. Reiss, D. J, B. Schwikowski: Predicting protein-peptide interactions via a network-based motif sampler. *Bioinformatics* 20 Supplement 1, I274-I282 (2004)

11. Bailey, T. L, C. Elkan: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Second International Conference on Intelligent Systems for Molecular Biology*, 28-36 (1994)

12. Puntervoll, P, R. Linding, C. Gemund, S. Chabanis-Davidson, M. Mattingsdal, S. Cameron, D. M. Martin, G. Ausiello, B. Brannetti, A. Costantini, F. Ferre, V. Maselli, A. Via, G. Cesareni, F. Diella, G. Superti-Furga, L. Wyrwicz, C. Ramu, C. McGuigan, R. Gudavalli, I. Letunic, P. Bork, L. Rychiewski, B. Kuster, M. Helmer-Citterich, W. N. Hunter, R. Aasland, T. J. Gibson: ELM server: a new resource for investigating short functional sites in modular eukaryotic proteins. *Nucleic Acids Res* 31 (13), 3625-3630 (2003)

13. Yaffe, M. B, G. G. Laparc, J. Lai, T. Obata, S. Volinia, L. C. Cantley: A motif-based profile scanning approach for genome-wide prediction of signaling pathways. *Nat Biotechnol* 19, 348-353 (2001)

14. Hulo, N, A. Bairoch, V. Bulliard, L. Cerutti, B. A. Cuche, E. de Castro, C. Lachaize, P. S. Langendijk-Genevaux, J. A. Sigrist: The 20 years of PROSITE. *Nucleic Acids Res,* 1-5 (2007)

15. Balla, S, V. Thapar, S. Verma, T. B. Luong, T. Faghri, C.-H. Huang, S. Rajasekaran, J. J. del Campo, J. H. Shinn, W. A. Mohler, M. W. Maciejewski, M.W. Gryk, B. Piccirillo, S. R. Schiller, M. Schiller: Minimotif miner: a tool for investigating protein function. *Nature Methods* 3 (3), 175-177 (2006)

16. Neduva, V, R. Linding, I. Su-Angrand, A. Stark, F. de Masi, T. J. Gibson, J. Lewis, L. Serrano, R. B. Russell: Systematic discoervy of new recognition peptides mediating protein interaction networks. *PLoS Biology* 3 (12), 2090-2099 (2005)

17. Bennett, S. P, C. G. Nevill-Manning, D. L. Brutlag: 3MOTIF: visualizing conserved protein sequence motifs in the protein structure database. *Bioinformatics* 19 (4), 541-542 (2003)

18. Li, H, J. Li: Discovery of stable and significant binding motif pairs from PDB complexes and protein interaction datasets. *Bioinformatics* 21 (3), 314-324 (2005)

19. Tan, S. H, W. Hugo, W. K. Sung, S. K. Ng: A correlated motif approach for finding short linear motifs from protein interaction networks. *BMC Bioinformatics* 7, 502 (2006)

20. Brazma, A, I. Jonassen, I. Eidhammer, D. Gilbert: Approaches to the automatic discovery of patterns in biosequences. Report Number 113, Department of Informatics, University of Bergen, Bergen, Norway, December (1995)

21. Jonassen, I, J. F. Collins, D. G. Higgins: Finding flexible patterns in unaligned protein sequences. *Protein Science* 4(8), 1587-1595 (1995)

22. Jonassen, I: Efficient discovery of conserved patterns using a pattern graph. *Comput Appl Biosci* 13, 509-522 (1997)

23. Jonassen, I, I. Eidhammer: Discovering patterns conserved in sets of related protein sequences. *Proc. of Norwegian Informatics Conference*, Tapir, Norway, 95-112 (1995)

24. Smith, R. F, T. F. Smith: Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. National Academy of Science, USA,* January, 118-122 (1990)

25. Zhong, W, G. Altun, R. Harrison, P. C. Tai, Y. Pan: Improved k-means clustering algorithm for exploring local protein sequence motifs representing common structural property. *IEEE Transactions on Nanobioscience* 4(3), 255-265 (2005)

26. Han, K.F, D. Baker: Global properties of the mapping between local amino acid sequence and local structure in proteins. *Proc Natl Acad Sci USA* 93**,** 5814-8 (1996)

27. Roytberg, M. A: A search for common patterns in many sequences. *CABIOS* 8(1), 57-64 (1992)

28. Schuler, G. D, S. F. Altschul, D. J. Lipman: A workbench for multiple alignment construction and analysis. *PROTEINS: Structure, Function, and Genetics* 9, 180-190 (1991)

29. Vingron, M, P. Argos: Motif recognition and alignment for many sequences by comparison of dot-matrices. *J Mol Biol* 218, 33-43 (1991)

30. Brazma, A, I. Jonassen, I. Eidhammer, E. Ukkonen: Relation patterns and their automatic discovery in biosequences. Report Number 135, Department of Informatics, University of Bergen, Bergen, Norway, June (1997)

31. Jonassen, I, I. Eidhammer, W. R. Taylor: Discovery of local packing motifs in protein structures. *PROTEINS: Structure, Function, and Genetics* 34, 206-219 (1999)

32. Jonassen, I, I. Eidhammer, D. Conklin, W. R. Taylor: Structure motif discovery and mining the PDB. *Proc. German Conference on Bioinformatics* (2000)

33. Rajasekaran, S: Algorithms for motif search. In: *Handbook of Computational Molecular Biology.* Eds: Aluru, S, Chapman & Hall/CRC Press, 37-1—37-21 (2006)

34. Rajasekaran, S, S. Balla, C.-H. Huang, V. Thapar, M. Gryk, M. Maciejewski, M. Schiller: High performance exact algorithms for motif search. *Journal of Clinical Monitoring and Computing*, 19(4-5), 319-328 (2005)

35. Rigoutsos, I, and A. Floratos: Motif discovery without alignment or enumeration. *Proc. RECOMB,* 221-227 (1998)

36. Arents, G, K. Moudrianakis: Topography of the histone octamer surface: repeating structural motifs utilized in the docking of nucleosomal DNA. *Proc Natl Acad Sci USA* 90, 10489-10493 (1993)

37. Rigoutsos, I, A. Floratos: Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics* 14(1), 55-67 (1998)

38. Davey, N. E, D. C. Shields, R. J. Edwards: SLiMDisc: short linear motif discovery, correcting for common evolutionary descent. *Nucleic Acids Res* 34(12), 3546-3554 (2006)

39. Horowitz, E, S. Sahni, S. Rajasekaran: *Computer Algorithms,* Silicon Press, 2008.

40. Ye, K, W. A. Kosters, P. IJzerman: An efficient, versatile and scalable pattern growth approach to mine frequent patterns in unaligned protein sequences. *Bioinformatics* 23(6), 687-693 (2007)

41. Agrawal, R, R. Srikant: Fast algorithms for mining association rules. *Proc. 20th International Conference on Very Large Data Bases (VLDB),* 487-499 (1994)

42. Keich, U, P. Pevzner: Finding motifs in the twilight zone. *Bioinformatics* 18, 1374-1381 (2002)

43. Buhler, J, M. Tompa: Finding motifs using random projections. *Proc. Fifth Annual International Conference on Computational Molecular Biology (RECOMB)*, April (2001)

44. Rajasekaran, S, S. Balla, C.-H. Huang: Exact algorithms for planted motif challenge problems. *Journal of Computational Biology* 12(8), 1117-1128 (2005)

45. Eskin, E, P. Pevzner: Finding composite regulatory patterns in DNA sequences. *Bioinformatics* S1, 354-363 (2002)

46. Price, A, S. Ramabhadran, P. Pevzner: Finding subtle motifs by branching from sample strings. *Bioinformatics* 1(1), 1-7 (2003)

47. Hertz, G, G. Stormo: Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15, 563-577 (1999)

48. Lawrence, C. E, A. A. Reilly: An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer

sequences. *Proteins: Structure, Function, and Genetics* 7, 41-51 (1990)

49. Martinez, H. M: An efficient method for finding repeats in molecular sequences. *Nucleic Acids Res* 11(13), 4629-4634 (1983)

50. Brazma, A, I. Jonassen, J. Vilo, E. Ukkonen: Predicting gene regulatory elements in silico on a genomic scale. *Genome Res* 15,1202-1215 (1998)

51. Galas, D. J, M. Eggert, M. S. Waterman: Rigorous pattern-recognition methods for DNA sequences: Analysis of promoter sequences from Escherichia coli. *J Mol Biol* 186(1), 117-128 (1985)

52. Sinha, S, M. Tompa: A statistical method for finding transcription factor binding sites. *Proc. Eighth International Conference on Intelligent Systems for Molecular Biology*, 344-354 (2000)

53. Staden, R: Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in the Biosciences* 5(4), 293-298 (1989)

54. Tompa, M, An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. *Proc. Seventh International Conference on Intelligent Systems for Molecular Biology*, 262-271 (1999)

55. van Helden, J, B. André, B., J. Collado-Vides: Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J Mol Biol* 281(5), 827-842 (1998)

56. Davila, J, S. Rajasekaran: Extending pattern branching to handle challenging instances. *Proc. 6th International Symposium on Bioinformatics and Bioengineering (BIBE),* 65-69 (2006)

57. Davila, J, S. Balla, S. Rajasekaran: Space and time efficient algorithms for planted motif search. *Proc. International Workshop on Bioinformatics Research and Applications (IWBRA),* Reading, UK, May (2006)

58. Pevzner, P, S.-H. Sze: Combinatorial approaches to finding subtle signals in DNA sequences. *Proc. Eighth International Conference on Intelligent Systems for Molecular Biology*, 269-278 (2000)

59. Davila, J, S. Balla, S. Rajasekaran: Fast and practical algorithms for planted (*l,d*) motif search. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 544-555 (2007)

60. Chin, F.Y.L, H. C. M. Leung: Voting algorithms for discovering long motifs. *Proceedings of the Third Asia-Pacific Bioinformatics Conference (APBC)*, 261-271 (2005)

61. Rocke, E, M. Tompa: An algorithm for finding novel gapped motifs in DNA sequences. *Proc. Second International Conference on Computational Molecular Biology (RECOMB)*, 228-233 (1998)

62. Sagot, M. F, Spelling approximate repeated or common motifs using a suffix tree. *Springer-Verlag LNCS* 1380, 111-127 (1998)

63. Adebiyi, E. F, M. Kaufmann: Extracting common motifs under the Levenshtein measure: theory and experimentation. *Proc. Workshop on Algorithms for Bioinformatics (WABI)*, Springer-Verlag LNCS 2452, 140-156 (2002)

64. Wijaya, E, K. Rajaraman, S.-M. Yiu, W.-K. Sung: Detection of generic spaced motifs using submotif pattern mining. *Bioinformatics* 23(12), 1476-1485 (2007)

65. Favorov, A. V, M. S. Gelfand, A. V. Gerasimova, D. A. Ravcheev, A. A. Mironov, V. J. Makeev: A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. *Bioinformatics* 21 (10), 2240-2245 (2005)

66. Carvalho, A. M, A. T. Freitas, A. L. Oliveira, M-F. Sagot: A highly scalable algorithm for the extraction of cis-regulatory regions. *Proc. Asia-Pacific Bioinformatics Conference (APBC),* 273-282 (2005)

Key Words: motif search, computational techniques, patterns identification, exact algorithms, approximation algorithms , review

Send correspondence to: Sanguthevar Rajasekaran, Dept. of CSE, University of Connecticut, Storrs, CT 06269-2155, Tel: 860-486-2428, Fax: 860-486-4817, E-mail: rajasek@engr.uconn.edu

http://www.bioscience.org/current/vol14.htm