Discovering Active Motifs in Sets of Related Protein Sequences and Using Them for Classification

Jason T. L. Wang* Thomas G. Marr† Dennis Shasha‡

Bruce Shapiro§ Gung-Wei Chirn¶

April 16, 1994

Abstract

We describe a method for discovering active motifs in a set of related protein sequences. The method is an automatic two step process: (1) find candidate motifs in a small sample of the sequences; (2) test whether these motifs are approximately present in all the sequences. To reduce the running time, we develop two optimization heuristics based on statistical estimation and pattern matching techniques. Experimental results obtained by running these algorithms on generated data and functionally related proteins demonstrate the good performance of the presented method compared with visual method of O'Farrell and Leopold. By combining the discovered motifs with an existing fingerprint technique, we develop a protein classifier. When we apply the classifier to the 698 groups of related proteins in the PROSITE catalog, it gives information that is complementary to the BLOCKS protein classifier of Henikoff and Henikoff. Thus, using our classifier in conjunction with theirs, one can obtain high confidence classifications (if BLOCKS and our classifier agree) or suggest a new hypothesis (if the two disagree).

*Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102.

†Cold Spring Harbor Laboratory, 100 Bungtown Road, Cold Spring Harbor, NY 11724.

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012.

§Image Processing Section, Laboratory of Mathematical Biology, Division of Cancer Biology and Diagnosis, National Cancer Institute, National Institutes of Health, Frederick, MD 21701.

¶Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102.

Introduction

Finding patterns of conserved amino acid residues in sets of sequences is an important problem in computational biology, particularly in the study of functionally related proteins [1, 2, 3, 4, 5, 6, 7, 8, 9]. Approaches to the general problem of finding such sequence motifs range from the development of special-purpose programs (e.g., [10, 11, 12, 13]) to using local similarities search programs (e.g., [14, 15, 16, 17, 18, 19]) to using multiple sequence alignment programs (e.g., [20, 21, 22, 23]).

This paper presents a new approach to *discovering* commonly occurring (or active) motifs in a set of related protein sequences. The structures of the motifs we wish to discover are regular expressions of the form $*S_1 * S_2 * \ldots$. The $S_1, S_2, \ldots$ are *segments* of a sequence, i.e., subsequences made up of consecutive letters, and $*$ represents a variable length don't care (VLDC). In matching the expression $*S_1 * S_2 * \ldots$ with a sequence S, the VLDCs may substitute for zero or more letters in S.

Example

Consider the set $\mathcal{S}$ of three sequences in Figure 1.

```
S1:  FEVHYDPMISEDENRLWYPVEPSVG
S2:  GWWRADVNHPTYDPAPFRMKENRAR
S3:  WRYDPMNSEDKTMTITLVGWNRLCD
```

Figure 1. The set $\mathcal{S}$ of three sequences.

Suppose only exactly coinciding segments occurring in at least two sequences and having lengths greater than 3 are considered as 'active.' Then $\mathcal{S}$ contains one active motif:

$$*S_1[5,8]* = *\text{YDPM}* \Longleftrightarrow *S_3[3,6]* = *\text{YDPM}*$$

where $V[x,y]$ is a segment of a sequence V from the xth to the yth letter inclusively. If motifs occurring in all the three sequences within one mutation are considered as active, i.e., one mismatch, insertion or deletion is allowed in matching a motif with a sequence, then $\mathcal{S}$ contains six active motifs:

$$\begin{array}{rc}
 & *S_1[4,7]* = *\text{HYDP}\,* \\
\Longleftrightarrow & *S_1[5,8]* = *\text{YDPM}\,* \\
\Longleftrightarrow & *S_1[13,16]* = *\text{ENRL}\,* \\
\Longleftrightarrow & *S_2[11,14]* = *\text{TYDP}\,* \\
\Longleftrightarrow & *S_2[12,15]* = *\text{YDPA}\,* \\
\Longleftrightarrow & *S_3[2,5]* = *\text{RYDP}\,* \\
\Longleftrightarrow & *S_3[3,6]* = *\text{YDPM}\,*
\end{array}$$

Suppose motifs occurring in at least two sequences within zero mutation and having the form $*X*Y*$ of total length greater than 6 are considered as active. Then $\mathcal{S}$ contains two active motifs:

$$
\begin{aligned}
& *S_1[5,8]*S_1[10,12]* = *\mathrm{YDPM}*\mathrm{SED}* \\
\Longleftrightarrow\quad & *S_1[5,8]*S_1[14,16]* = *\mathrm{YDPM}*\mathrm{NRL}* \\
\Longleftrightarrow\quad & *S_3[3,6]*S_3[8,10]* = *\mathrm{YDPM}*\mathrm{SED}* \\
\Longleftrightarrow\quad & *S_3[3,6]*S_3[21,23]* = *\mathrm{YDPM}*\mathrm{NRL}*
\end{aligned}
$$

End of Example

To discover such active motifs in a set of sequences, our overall strategy is first to find candidate segments among a small sample (e.g., YDPM and SED in the last example) and then to combine the segments into candidate motifs (e.g. *YDPM*SED* in the last example) which we check against the entire set.

A number of previous techniques may be used to locate the active motifs. These techniques work based on either one of the following two approaches: (i) a multiple alignment of the sequences as a whole, and (ii) a search for local similar segments (or similarities) in the set. The first approach is useful when entire sequences in the set are similar. However, when the sequences have only short regions of local similarities, this approach is inapplicable. Published algorithms of the second approach work effectively when the similar segments meet some constraints, such as that they occur in a predetermined number of sequences in the set [24], they differ by mismatches, but not by insertions/deletions [15], or they are situated at almost the same distance from the start of the sequences [23]. In contrast to these algorithms, our method can find active motifs composed of nonconsecutive segments separated by variable length don't cares without prior knowledge of their structures, positions, or occurrence frequency.

We have implemented the proposed method into a system, called DISCOVER. By combining the method with a previously published fingerprint technique [25], we have developed a protein classifier, called CLASSIFY. We applied CLASSIFY to all 698 groups of related proteins documented in the PROSITE catalog [26]. It does as well as the BLOCKS database of Henikoff and Henikoff [13] in terms of the number of correct classifications (assuming the classifications in PROSITE are all correct), but misclassifies different sequences. Thus, using CLASSIFY in conjunction with the BLOCKS database either gives high confidence to the classification (if the two agree) or suggests a new family to examine (if the two disagree).

Methods

The DISCOVER and CLASSIFY systems

DISCOVER takes a set of related proteins and produces a collection of active motifs in the set. CLASSIFY accepts a query protein and displays a PROSITE group to which the protein should belong. These systems can be executed either manually using user-specified parameters or automatically using parameters determined by the systems.

The programs are written in the C programming language and are compiled for IBM-compatible personal computers (DOS version), DEC systems (DEC-ULTRIX version) and Sun SPARC workstations (SPARC-UNIX version). The DOS and DEC-ULTRIX versions are available on a floppy disk upon request. The SPARC-UNIX version is accessible via electronic mail. For proper use of the server, send a word HELP on the subject line to discover-classify@cis.njit.edu. If you do not receive a satisfactory response, please contact us directly at jason@cis.njit.edu or shasha@cs.nyu.edu.

Terminology

Let $\mathcal{S}$ be a set of sequences. The occurrence number (or activity) of a motif is the number of sequences in $\mathcal{S}$ that match the motif within the allowed number of mutations. We say the occurrence number of a motif P with respect to mutation i and set $\mathcal{S}$, denoted $occurrence\_no^i_{\mathcal{S}}(P)$, is k if $*P*$ matches k sequences in $\mathcal{S}$ within at most i mutations, i.e., the k sequences contain P within i mutations. For example, in Figure 1, $occurrence\_no^0_{\mathcal{S}}(*\mathrm{YDPM}*) = 2$ and $occurrence\_no^1_{\mathcal{S}}(*\mathrm{HYDP}*) = occurrence\_no^1_{\mathcal{S}}(*\mathrm{YDPM}*)$ $= occurrence\_no^1_{\mathcal{S}}(*\mathrm{ENRL}*) = occurrence\_no^1_{\mathcal{S}}(*\mathrm{TYDP}*) = occurrence\_no^1_{\mathcal{S}}(*\mathrm{YDPA}*)$ $= occurrence\_no^1_{\mathcal{S}}(*\mathrm{RYDP}*) = 3$.

Given a set $\mathcal{S}$, DISCOVER can find all the active motifs P where P is within the allowed Mut mutations of at least $Occur$ sequences in $\mathcal{S}$ and $|P| \geq Length$, where $|P|$ represents the number of the non-VLDC letters in the motif P. (Mut, $Occur$, $Length$ and the form of P are user-specified parameters.) The basic subroutine in DISCOVER is to match a given motif against a given sequence after an optimal substitution for the VLDCs in the motif. For example, in matching the motif *TQI* with a sequence WYALTIHKR, the first asterisk would substitute for WYAL and the second asterisk would substitute for HKR. The motif is within one mutation of the sequence since Q is deleted. The length of the motif is three.[1]

Discovery algorithm

Our algorithm consists of two phases: (1) find candidate segments among a small sample $\mathcal{A}$ of the sequences; (2) combine the segments to form candidate motifs and evaluate the activity of the motifs in all of $\mathcal{S}$ to determine which motifs satisfy the specified requirements.

Phase (1) consists of two subphases. In subphase A, we construct a *generalized suffix tree* [28] (GST) for the sample of sequences. A suffix tree is a trie-like data structure that compactly represents a string by collapsing a series of nodes having one child to a single node whose parent edge is associated with a string [29, 30]. A GST is an extension of the suffix tree, designed for representing a set of strings. Each suffix of a string is represented by a leaf in the GST. Each leaf is associated with an index i. Let $subtree(v)$ be the subtree rooted at a non-leaf node v. We use $count(v)$ to represent the number of different indexes associated with the leaves in $subtree(v)$. The edges are labeled with character strings such that the concatenation of the edge labels on the path from the root to the leaf with index i is a suffix of the ith string in the set. See Figure 2 for an example (the node labeled with a 1 above the leaf MTRM is

[1] Given a regular expression motif P and sequence S, one can determine whether P is within Mut mutations of S in $O(Mut \times |S|)$ time when $O(|P|) = O(\log |S|)$ [27].

an example of the result of a collapsing). The GST can be constructed asymptotically in $O(n)$ time and space where n is the total length of all sequences in the sample $\mathcal{A}$.

In subphase B, we traverse the GST constructed in subphase A to find all segments (i.e., all prefixes of strings labeled on root-to-leaf paths) that satisfy the length minimum. If the pattern specified by the user has the form $*X*$, then the length minimum is simply the specified minimum length of the pattern. If the pattern specified by the user has the form $*X_1 * X_2*$, we find all the segments V_1, V_2 where at least one of the V_i, $1 \leq i \leq 2$, is (larger than or equal to) half of the specified length and the sum of their lengths satisfies the length requirement. If the user-specified pattern has the form $*X_1 * X_2 * \ldots * X_k*$, we find the segments $V_1, V_2, \ldots, V_k$ where at least one of the V_i, $1 \leq i \leq k$, is (larger than or equal to) $1/k$th of the specified length and the sum of the lengths of all these segments satisfies the length requirement.

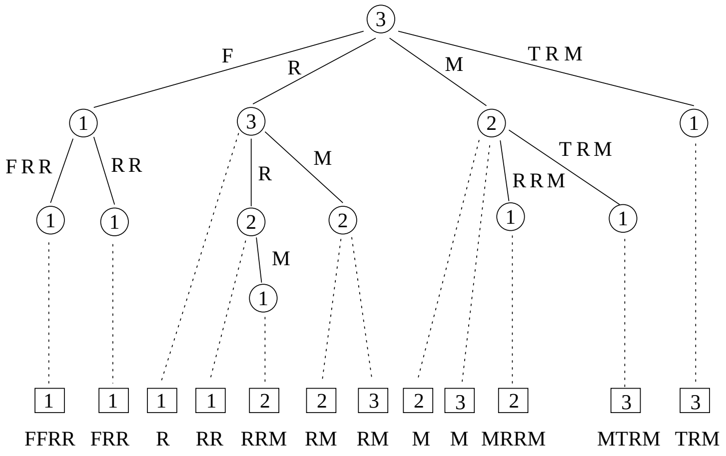

Figure 2. The GST for a sample $\mathcal{A}$ = {FFRR, MRRM, MTRM}. Leaves are represented as rectangles, labeled with the indexes. Non-leaf nodes are represented as circles, labeled with the *count* values. The suffix corresponding to a leaf is shown below the leaf. Note that the suffixes RM and M appear in two strings and hence appear twice in the leaves.

Phase (2) also has two subphases. In subphase A, we evaluate the activity of the candidate motifs and rank them from highest to lowest according to their occurrence numbers on the sample with respect to mutation Mut. If interesting motifs are of the form $*X_1 * X_2 * \ldots$, we consider all possible combinations $V_1, V_2, \ldots$ of the segments obtained in phase (1) that meet the length requirement and match $*V_1 * V_2 * \ldots$ with the sequences in the sample. Subphase B evaluates the most likely candidate motifs found in subphase A with respect to the entire set.[2]

Optimization heuristics

In phase (2) of the discovery algorithm, we compare only the most likely candidate motifs with the entire set. The main question from an optimization point of view is which candidates to compare. Our strategy is as follows.

[2] As our experimental results show later, screening out those unlikely candidate motifs in the first subphase saves significant time in the overall computation.

We use *simple random sampling without replacement* [31] to select sample sequences from the set. Consider a candidate motif P. Let M (a, respectively) denote the number of sequences in the entire set $\mathcal{S}$ (the sample $\mathcal{A}$, respectively) that contain P within the allowed number of mutations. Let N be the set size and n the sample size; $F = M/N$ and $f = a/n$. Then, with probability = 99%, F is in the interval $(\hat{F}_L, \hat{F}_U)$ where

$$\hat{F}_L = f - (t\sqrt{\frac{N-n}{N-1}}\sqrt{\frac{f(1-f)}{n}} + \frac{1}{2n}),$$

$$\hat{F}_U = f + (t\sqrt{\frac{N-n}{N-1}}\sqrt{\frac{f(1-f)}{n}} + \frac{1}{2n}).$$

The symbol t is the value of the normal deviate corresponding to the desired confidence probability. When the probability = 99%, $t = 2.58$ [31]. The values of N, n are given; f, a can be obtained from subphase A of phase (2). Thus, if the estimator $(\hat{F}_U \times N) < Occur$ for the candidate motif P, then with probability $\geq 99\%$, P won't be an active motif satisfying the specified requirements. We therefore discard it.

The second optimization heuristic we implemented is to eliminate the redundant calculation of occurrence numbers. Observe that the most expensive operation in our discovery algorithm is to find the occurrence number of a motif with respect to the entire set, since that entails matching the motif against all sequences. We say $*U_1 * \ldots * U_m*$ is a *subpattern* of $*V_1 * \ldots * V_m*$ if U_i is a subsegment of V_i, for $1 \leq i \leq m$.

One can observe that if motif P is a subpattern of motif P', then $occurrence\_no^k_{\mathcal{S}}(P) \geq occurrence\_no^k_{\mathcal{S}}(P')$ for any mutation parameter k. Thus, if P' is in the final output set, then we need not bother matching P against sequences in $\mathcal{S}$, since it will be too. If P is not in the final output set, then P' won't be either, since its occurrence number will be even lower.

To illustrate how the above two optimization heuristics are incorporated into the discovery algorithm, consider finding the motifs of the form $*X * Y*$ with total length greater than or equal to 5. We begin by enumerating segments of length 3 in the generalized suffix tree (GST). Let $string(v)$ be the string on the edge labels from the root to v. If the above statistical estimator tells us that the combination of a segment $string(u_1)$ of length 3 with another segment $string(u_2)$ does not yield an active enough motif satisfying the specified Mut and $Occur$ requirements, then we eliminate the pair $string(v_1)$ and $string(v_2)$ from consideration, where v_1 and v_2 are descendants of u_1 and u_2, respectively. Similar pruning operations can be applied when enumerating longer segments in the GST.

Classification algorithm

We applied the discovery algorithm to all 698 groups of related proteins documented in the PROSITE catalog v. 11.0 keyed to the SWISS-PROT protein sequence databank version 27 [32]. We selected 70% of the sequences in each group at random to serve as a training sample. We then processed the training sequences in two ways:

- Find 50 characteristic motifs from the training sample of each group. The motifs are the length 4 segments having the highest occurrence numbers with zero mutations. When there are ties for occurrence numbers with respect to zero mutations, we break the ties by considering occurrence numbers with respect to one mutation.[3] To reduce the effect made by 'chance motifs,' we associate each characteristic motif with a weight based on Zipf's Law [33]. If a motif occurs in m groups, its weight is assigned as $\log_2\lceil(M/m)\rceil$, where M is the total number of groups, 698 in our case.

- Hash the training sequences using the gapped fingerprint technique [25].

When classifying a query sequence T, we first compare T with all the characteristic motifs. After comparison, each group obtains a raw score, which equals the sum of the weights of the group's characteristic motifs occurring in T. The raw score for a group is normalized by dividing it by the total weight of all the characteristic motifs in that group and multiplying by 100. The highest-scoring group is then displayed as the result of the classification provided that its score is greater than an experimentally determined threshold. (In the study presented here, the threshold was set to 20.) Otherwise we proceed to the second phase.

In the second phase, we hash T, using the same hash function as the one used for the training sequences. The group containing sequences with the highest vote is displayed as the result of the classification. If two sequences have the same highest vote, the shorter one is favored.

Results

Performance analysis of the discovery algorithm

We carried out a series of experiments to evaluate the effectiveness (measured by accuracy with respect to exhaustive search) and speed of the proposed discovery algorithm. The programs were written in C and run on a Sun SPARC workstation under the SUN operating system version 4.1.2. The data was a set of 150 kinase sequences, with the lengths ranging from 220 residues to 2500 residues. The active motifs of interest had the form $*X*Y*$.

The metric used to evaluate the effectiveness of our algorithm is the *hit ratio*, defined as $(NumDiscovered/TotalNum) \times 100\%$ where $NumDiscovered$ is the number of active motifs discovered by our algorithm. $TotalNum$ is the number of active motifs obtained from the exhaustive search method. By exhaustive search, we mean selecting as candidates all combinations of the segment pairs V_1, V_2 appearing in the set that satisfy length constraints.[4]

The experimental results indicated that the effectiveness of the discovery algorithm depends on both the sample size and the number of mutations allowed during searching. For example, when $Mut = 0$, DISCOVER is nearly as accurate as exhaustive search provided $SizeRatio \geq 0.2$. ($SizeRatio$ represents the ratio between the sample size and the set size.) When $Mut = 1$, the hit ratio reaches 80% provided the $SizeRatio \geq 0.4$.

[3] We chose this length and this number of motifs because this seems to give good results. These decisions can be changed easily and are compile-time parameters of our system.

[4] We could conceivably have generated approximately occurring motifs that never appear in the set yet satisfy the Mut and $Occur$ constraints, but we considered this "super-exhaustive" approach too time-consuming and of dubious value.

We next compared the running times of the algorithm for the $Mut = 1$ case. We found that DISCOVER runs significantly faster than the exhaustive search method. Even with $SizeRatio = 0.8$, in which case DISCOVER achieves nearly 100% hit ratio, it is more than 10 times faster than exhaustive search.[5] We also observed that the optimization heuristics reduce the running time substantially. Together, they can speed up the discovery algorithm by a factor of nearly 100.

Discovery of active motifs in protein families

In this experiment we examined three protein families (cyclin, ras and kinase) to see whether the motifs obtained correspond to those shown in previous studies which used other methods. The cyclin family contained 47 protein sequences, with the lengths ranging from 190 residues to 780 residues. The ras family contained 149 protein sequences, with the lengths ranging from 35 residues to 3079 residues. The kinase family contained 1077 protein sequences, with the lengths ranging from 10 residues to 2938 residues.

Tables 1, 2 and 3 show the motifs of the form $*X*$ found by DISCOVER and their occurrence numbers with respect to mutation i, $0 \leq i \leq 4$, for the three protein families, respectively. The tables show, for each length of motifs, the top one (or two) most active motifs discovered in each family. The activity of a motif in a family is ranked in terms of its occurrence number with respect to mutation 0.

Motifs found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
LQL	27	47	47	47	47
QLV	26	47	47	47	47
QLVG	24	35	47	47	47
KYEE	20	42	47	47	47
LQLVG	19	29	43	47	47
ASKYEE	13	25	35	46	47
KLQLVG	13	22	33	47	47
IASKYEE	9	20	29	36	47

Table 1. Motifs discovered for the cyclin family ($SizeRatio =$ 20%). For each length of motifs, only the top one (or two) most active ones discovered are shown in the table.

[5] When the sample is this large, the two segments V_1, V_2 of nearly every interesting motif appear in the sample. Our algorithm works by enumerating all promising segment pairs in the sample, and therefore can find all the interesting motifs.

Motifs found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
TAG	106	147	149	149	149
DTAG	99	125	148	149	149
DTAGQ	92	104	134	148	149
LVGNK	62	103	138	149	149
DTAGQE	90	101	110	146	149
WDTAGQE	50	90	101	111	147
GVGKSALT	41	45	61	88	130
YDPTIEDSY	38	41	42	47	76

Table 2. Motifs discovered for the ras family ($SizeRatio = 20\%$). For each length of motifs, only the top one (or two) most active ones discovered are shown in the table.

Motifs found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
DFG	338	1018	1076	1077	1077
ELL	331	1048	1075	1077	1077
HRDL	174	484	1034	1077	1077
DFGL	166	567	1062	1076	1077
DFGLA	127	257	844	1070	1076
FGLAR	97	183	817	1070	1077
DFGLAR	97	146	367	967	1072
RDLAARN	67	79	124	515	1039

Table 3. Motifs discovered for the kinase family ($SizeRatio = 5\%$). For each length of motifs, only the top one (or two) most active ones discovered are shown in the table.

From these tables, it can be seen that shorter motifs tend to have higher occurrence numbers. The occurrence frequency of motifs is family dependent. In the ras family, for example, there is a very active segment DTAGQE, which appears in more than 60% proteins in the family. On the other hand, in the kinase family, the most active segment of the same length, DFGLAR, appears in less than 10% proteins in the family.

There are also motifs composed of segments appearing nonconsecutively in the protein sequences. Table 4 shows several active motifs composed of 2 nonconsecutive segments in the ras family.

Motifs composed of 2 nonconsecutive segments found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
∗DTAGQE∗LVGNK∗	52	93	97	104	112
∗GGVGKSALT∗LVGNK∗	29	40	43	55	63
∗YDPTIEDSY∗LVGNK∗	29	40	40	44	58
∗YDPTIEDSY∗DTAGQE∗	31	36	41	46	57
∗GGVGKSALT∗DTAGQE∗	28	40	44	61	82

Table 4. Occurrence numbers for the active motifs composed of 2 nonconsecutive segments in the ras family.

It is worth pointing out that the motifs discovered in the cyclin and ras sequences are a superset of those found manually by O'Farrell and Leopold [34]. The kinase sequence motifs that we were able to detect overlap with the sequence motifs described in [12, 35, 36].

Classification proteins in the PROSITE groups

In this experiment we examined the effectiveness of the proposed classification algorithm by applying it to the 698 groups in the PROSITE catalog. The groups comprise more than 15,000 proteins. Currently, the best classifier for these proteins is the BLOCKS database developed by Henikoff and Henikoff in [13]. Their algorithm associates each group with a set of blocks, where a block comprises ungapped aligned regions extracted from the sequences in the group. To classify a query sequence, BLOCKS matches the query against all the blocks and displays a collection of groups, ranked based on their relevance to the query.

In contrast to BLOCKS, we selected 70% of the sequences in each group at random to serve as a training sample and processed the training sequences by finding their characteristic motifs and hashing them as described in the Methods section. We then checked whether the remaining 30% (the test sequences) were classified correctly according to our CLASSIFY algorithm and according to the BLOCKS method. Note that this experiment favors BLOCKS since the blocks database is built from all sequences including the 30% that our method treats as unknowns.

Table 5 summarizes the classification results. A test sequence was classified correctly by BLOCKS (respectively, CLASSIFY) if its group was ranked highest by BLOCKS (respectively, CLASSIFY). The table also shows the results when the two methods agreed (i.e., the highest ranked group returned by both of them was the same) and disagreed on their rankings.[6]

[6] A separate finding from the experiment was that about 30% of the test sequences went to the second phase of our classification algorithm (cf. the Methods section).

Classification results	Percentage of the test sequences
BLOCKS was correct	93.5%
CLASSIFY was correct	92.3%
BLOCKS and CLASSIFY agreed & both were correct	86.7%
BLOCKS and CLASSIFY agreed & both were wrong	0.5%
BLOCKS and CLASSIFY disagreed & BLOCKS was correct	7.3%
BLOCKS and CLASSIFY disagreed & CLASSIFY was correct	4.8%
BLOCKS and CLASSIFY disagreed & both were wrong	0.7%

Table 5. Comparison between CLASSIFY and BLOCKS in classifying proteins in the PROSITE groups. (Note: The last five percentages in the table add up to 100%.)

Thus if BLOCKS and CLASSIFY agree, the classification has a high likelihood of being correct. Specifically, the correct agreed-upon classifications divided by the total agreed-upon classification is 86.7%/(86.7% + 0.5%) = 99.4%. On the other hand, if BLOCKS and CLASSIFY disagree, then the likelihood that one is right is (7.3% + 4.8%)/(7.3% + 4.8% + 0.7%) = 94.5%.

We also compared both tools' capability to detect distantly related sequences. As a specific example, we looked at the G-protein coupled receptor family (AC# PS00237) in SWISS-PROT 27. This family was chosen because Henikoff and Henikoff [13] used the family to evaluate the effectiveness of the BLOCKS database. We used the same set of 16 sequences of this family as used in their paper in the test. These 16 were not in our training set.

Table 6 shows the search results returned by CLASSIFY and the BLOCKS server, displaying only the blocks getting the highest score. It can be seen that all the 16 sequences were classified into their family by our tool. For the BLOCKS server, 15 of the 16 sequences were classified correctly. The highest scoring group (block) for the human thromboxane A2 receptor (TA2R$HUMAN) was BL00231E.

SWISS-PROT ID	group hit by CLASSIFY	group hit by BLOCKS
UL33$HCMVA	PS00237	BL00237B
CANR$HUMAN	PS00237	BL00237B
CANR$RAT	PS00237	BL00237B
ET1R$BOVIN	PS00237	BL00237B
ETBR$RAT	PS00237	BL00237A
FSHR$RAT	PS00237	BL00237B
LSHR$PIG	PS00237	BL00237B
LSHR$RAT	PS00237	BL00237B
GRPR$MOUSE	PS00237	BL00237A
US28$HCMVA	PS00237	BL00237B
TSHR$CANFA	PS00237	BL00237B
TSHR$RAT	PS00237	BL00237B
TSHR$HUMAN	PS00237	BL00237B
NTR$RAT	PS00237	BL00237A
PAFR$CAVPO	PS00237	BL00237B
TA2R$HUMAN	PS00237	BL00231E

Table 6. Comparison between CLASSIFY and BLOCKS in detecting distantly related sequences in the G-protein coupled receptor family (AC# PS00237) in SWISS-PROT 27. (Note: The A and B in the BLOCKS data indicate which characteristic block the protein hit with rank 1.)

Discussion

Comparing DISCOVER with previous algorithms for multiple sequence alignment and local similarities search

One principal advantage of DISCOVER over multiple alignment algorithms [20, 21, 23, 37, 38] is that it can find short regions of local similarities in a set of sequences. The recently published MOTIF program [11], implemented based on multiple alignment techniques, can also discover short regions (in some cases, only one amino acid) separated by a fixed number of distances (i.e., fixed length don't cares). However, MOTIF is unable to locate similar segments separated by variable length don't cares.

There are a number of published techniques for finding local similarities in multiple sequences. Queen *et al.* [14], for example, gave a method for searching for exact matches in the sequences using special tables. Sobel and Martinez [17] presented techniques that search for exact matches and then link them into chains. Bacon and Anderson [15] showed a method able to find segments allowing mismatches. Taylor [18] compared the sequences with the aid of templates. Krishnan *et al.* [16] and Vihinen [19] sought for similar segments using overlapped dot-matrices. None of these methods, however, can find 'weak similarities' [24], i.e., similar segments that (i) can contain no regions of exact match, (ii) can differ in both substitutions and insertions/deletions, and (iii) can be situated at arbitrary positions on the sequences in the set.

The only method that can find the weak similarities in a set of sequences, as DISCOVER does, is the one recently published by Roytberg [24]. Roytberg considered a segment of a sequence to be 'fundamental' if every sequence contains at least one segment similar to it. His method selects one sequence as 'basic' and searches for the fundamental segments on it. Then the segments common to all the sequences are reconstructed using the fundamental segments found on the basic sequence. To find the fundamental segments, the basic sequence is compared successively with all the other sequences. The result of each comparison is a collection of all the segments of a basic sequence that are similar to any segment of the other sequence. After all these comparisons have been made, the fundamental segments of the basic sequence are constructed by intersections of the segments revealed by pairwise comparisons. In this way, one can find all the weak similarities in the set.

Although both DISCOVER and Roytberg's method can locate weak similarities, they differ in three significant ways. First, Roytberg's technique requires the similar segments to occur in all the sequences in the given set or in a large portion f of the set. The value for the parameter f must be known in advance. In contrast, DISCOVER can find the similar segments without prior knowledge of their occurrence frequency. Second, by employing the algorithm for approximate regular expression matching, DISCOVER can find active motifs composed of nonconsecutive segments. Third, DISCOVER uses optimizations that make it suitable even when the given set is large.[7]

We implemented Roytberg's method and ran it on some sets of 10 sequences, where the similar segments can be found in all the sequences. The lengths of the sequences ranged from 70 letters to 170 letters. The data were made up by choosing the protein sequences from the ras family and by using a random character

[7]The reader may refer to [39] where we discuss several extensions of our discovery algorithm to deal with very large databases.

generator. The results obtained from these data were consistent. It was observed that the behavior of Roytberg's method is sensitive to both the choice of basic sequence and the algorithms employed in doing pairwise comparison.

For example, when using the exhaustive search algorithm to find all weak similarities in two sequences, with a fixed number (say, 3) mutations allowed, and trying out every sequence as basic, in searching for the active segments occurring in all 10 sequences, Roytberg's method returned optimal solutions, as did DISCOVER. On the other hand, using the algorithms suggested in Roytberg's paper (e.g., the Goad and Kanehisa's algorithm [40] with the score measures: +1 for each character match, -3 for each mismatch and -3 for each deletion from a sequence) the Roytberg method returned fewer similar segments than DISCOVER, even choosing all sequences as basic.

When searching the similar segments of lengths between 4 and 10 letters with 3 mutations allowed, Roytberg's method (with the exhaustive search algorithm) required 12 minutes on average. On the other hand, DISCOVER only required 5 minutes, even using all the 10 sequences as the sample.

Comparing CLASSIFY with previous algorithms for classifying proteins and detecting distant similarities

There are a number of methods published for classifying proteins and/or detecting distant similarities. One common way to do this is via profile analysis [10, 18, 41]. In contrast to our method, which uses the most frequently occurring ungapped segments and fingerprints, profiles are obtained by global multiple alignments including gapped regions. Each profile is a position-specific scoring table, created by aligning a group of related sequences. Each column of the alignment is converted to a column of a matrix representing the occurrence frequency of an amino acid. When determining the relevance of a query sequence to a group, one compares the sequence to all the profiles in the database.

Another technique for detecting distantly related sequences is via testing AACC (amino acid class covering) patterns [12]. AACC patterns are generated by clustering the pairwise similarity scores among a set of related sequences to build a binary tree. The tree is then reduced in a stepwise manner by progressively replacing the node connecting the two most similar termini by one common pattern until only a single common root pattern remains. Like the characteristic motifs we use, the AACC patterns serve as the representations of the related sequences. However, unlike our approach, which uses the protein groups documented in the PROSITE catalog, the AACC algorithm automates the process of making groups by clustering an entire database into coverings. Since such a clustering procedure can be tricky for distantly related sequences, a group in the PROSITE catalog may correspond to several coverings generated by the AACC method.

The recently published BLOCKS database [13, 42], with which we compared our method in the Results section, also used the protein groups documented in the PROSITE catalog. Each group corresponds to a set of blocks obtained from ungapped aligned regions discovered by the MOTIF program [11]. A best set of blocks is then selected using a heuristic algorithm called MOTOMAT. All the selected blocks are finally calibrated and concatenated into the database.

Although both BLOCKS and CLASSIFY exploit the group information and documentation with PROSITE, they differ in their fundamental methods. First and most important, blocks are built based on multiple local alignments, whereas CLASSIFY generates characteristic motifs by running approximate regular expression matching algorithms. Second, each group is scored as a single unit in our system. However, there are typically multiple blocks for a group that are scored independently. Third, BLOCKS displays a list of groups, ranked based on their relevance to the query sequence, whereas CLASSIFY shows only one group to which the sequence should belong. Finally, in displaying the output, BLOCKS shows the alignment between a hit block and the query sequence, whereas CLASSIFY shows the characteristic motifs hitting the query or the training sequence with the highest vote (the output not shown here). These two displays can be useful at different times.

As indicated in our experimental results, the use of our system complements the standard search and classification techniques. We expect that the systems can be used in many other applications as well. For example, DISCOVER may help establish DNA and protein super-families – two families may be closely related if they have many active motifs in common. The CLASSIFY algorithm can also be used to detect whether or not a sequence is a member of a known DNA family.

Acknowledgments

The authors thank the anonymous referees and the editor, Dr. Richard Roberts, for their encouragement and thoughtful suggestions. We also thank Jorja Henikoff for offering programs to generate PROSITE groups; Amos Bairoch for providing software information available on the Internet; and to Gadi Landau, Udi Manber, Sun Wu and Kaizhong Zhang for their hints regarding approximate string and regular expression matching. This work was supported, in part, by the National Science Foundation under grants IRI-8901699, CCR-9103953, IRI-9224601 and IRI-9224602, by the Office of Naval Research under grants N00014-90-J-1110, N00014-91-J-1472 and N00014-92-J-1719, by the New Jersey Institute of Technology under Grant No. 421280 and by a grant from the AT&T Foundation. T. G. M was supported by the Department of Energy under grant DE-FG02-91ER61190 and by the National Institutes of Health under grant 1RO1-HG0020301A1.

References

[1] Bashford, D., Chothia, C. and Lesk, A. M. (1987) *J. Mol. Biol.*, **196**, 199–215.

[2] Caserta, M., Zacharias, W., Nwankwo, D., Wilson, G. G. and Wells, R. D. (1987) *J. Biol. Chem.*, **262**, 4770–4777.

[3] Hunter, T. (1987) *Cell*, **50**, 823–829.

[4] Som, S., Bhagwat, A. S. and Friedman, S. (1987) *Nucleic Acids Research*, **15**, 313–332.

[5] Sznyter, L. A., Slatko, B., Moran, L., O'Donnell, K. H. and Brooks, J. E. (1987) *Nucleic Acids Research*, **15**, 8249–8266.

[6] Doolittle, R. F., Feng, D.-F., Johnson, M. S. and McClure, M. A. (1989) *Q. Rev. Biol.*, **64**, 1–30.

[7] Lipton, R. J., Marr, T. G. and Welsh, J. D. (1989) *Proceedings of the IEEE*, **77**, 1056–1060.

[8] Posfai, J., Bahwat, A. S., Posfai, G. and Roberts, R. J. (1989) *Nucleic Acids Research*, **17**, 2421–2435.

[9] Romisch, K., Webb, J., Herz, J., Prehn, S., Frank, R., Vingron, M. and Dobberstein, B. (1989) *Nature (London)*, **340**, 478–482.

[10] Gribskov, M., McLachlan, A. D. and Eisenberg, D. (1987) *Proc. Natl. Acad. Sci. USA*, **84**, 4355–4358.

[11] Smith, H. O., Annau, T. M. and Chandrasegaran, S. (1990) *Proc. Natl. Acad. Sci. USA*, **87**, 826–830.

[12] Smith, R. F. and Smith, T. F. (1990) *Proc. Natl. Acad. Sci. USA*, **87**, 118–122.

[13] Henikoff, S. and Henikoff, J. G. (1991) *Nucleic Acids Research*, **19**, 6565–6572.

[14] Queen, C., Wegman, M. N. and Korn, L. J. (1982) *Nucleic Acids Research*, **10**, 449–456.

[15] Bacon, D. J. and Anderson, W. J. (1986) *Journal of Molecular Biology*, **191**, 153–161.

[16] Krishnan, G., Kaul, R. K. and Jagadeeswaran, P. (1986) *Nucleic Acids Research*, **14**, 543–550.

[17] Sobel, E. and Martinez, H. M. (1986) *Nucleic Acids Research*, **14**, 363–374.

[18] Taylor, W. R. (1986) *Journal of Molecular Biology*, **188**, 233–258.

[19] Vihinen, M. (1988) *Computer Applications in the Biosciences*, **4**, 89–92.

[20] Johnson, M. J. and Doolittle, R. F. (1986) *J. Mol. Evol.*, **23**, 267–277.

[21] Waterman, M. S. (1986) *Nucleic Acids Research*, **14**, 9095–9102.

[22] Lipman, D. J., Altschul, S. F. and Kececioglu, J. D. (1989) *Proc. Natl. Acad. Sci. USA*, 4412–4415.

[23] Vingron, M. and Argos, P. (1989) *Computer Applications in the Biosciences*, **5**, 115–122.

[24] Roytberg, M. A. (1992) *Computer Applications in the Biosciences*, **8**, 57–64.

[25] Califano, A. and Rigoutsos, I. (1993) *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, Bethesda, MD.

[26] Bairoch, A. (1992) *Nucleic Acids Research*, **20**, 2013–2018.

[27] Wu, S. and Manber, U. (1992) *Communications of the ACM*, **35**, 83–91.

[28] Hui, L. C. K. (1992) In Apostolico, A., Crochemore, M., Galil, Z. and Manber, U. (ed.), Combinatorial Pattern Matching, Lecture Notes in Computer Science, Springer-Verlag, **644**, 230–243.

[29] McCreight, E. M. (1976) *Journal of the ACM*, **23**, 262–272.

[30] Landau, G. M. and Vishkin, U. (1989) *Journal of Algorithms*, **10**, 157–169.

[31] Cochran, W. G. (1977) *Sampling Techniques*, Wiley.

[32] Bairoch, A. and Boeckmann, B. (1992) *Nucleic Acids Research*, **20**, 2019–2022.

[33] Zipf, G. K. (1949) *Human Behavior and the Principle of Least Effort*, Addison Wesley, Reading.

[34] O'Farrell, P. and Leopold, P. (1991) *Cold Spring Harbor Symposia on Quantitative Biology*, Vol. LVI, Chapter A consensus of cyclin sequences reveals homology with the ras oncogene, Cold Spring Harbor Press, Cold Spring Harbor.

[35] Hanks, S. K., Quinn, A. M. and Hunter, T. (1988) *Science*, **241**, 42–52.

[36] Smith, R. F. and Smith, T. F. (1989) *J. Virol.*, **63**, 450–455.

[37] Waterman, M. S., Arratia, R. and Galas, D. J. (1984) *Bull. Math. Biol.*, **46**, 515–527.

[38] Bains, W. (1986) *Nucleic Acids Research*, **14**, 159–177.

[39] Wang, J. T. L., Chirn, G.-W., Marr, T. G., Shapiro, B., Shasha, D. and Zhang, K. (1994) *Proceedings of the 1994 ACM SIGMOD International Conference on the Management of Data*, Minneapolis, Minnesota.

[40] Goad, W. B. and Kanehisa, M. I. (1982) *Nucleic Acids Research*, **10**, 247–263.

[41] Staden, R. (1990) *Meth. Enzymol.*, **183**, 193–211.

[42] Wallace, J. C. and Henikoff, S. (1992) *Computer Applications in the Biosciences*, **8**, 249–254.