An Extension and Novel Solution to the (*l,d*)–Motif Challenge Problem

Mark P Styczynski[1] `marksty@mit.edu`
Kyle L Jensen[1] `kljensen@mit.edu`
Isidore Rigoutsos[2] `rigoutso@us.ibm.com`
Gregory N Stephanopoulos[1] `gregstep@mit.edu`

[1] Department of Chemical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
[2] Bioinformatics and Pattern Discovery Group, IBM Thomas J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, USA

Abstract

The (l,d)–motif challenge problem, as introduced by Pevzner and Sze [12], is a mathematical abstraction of the DNA functional site discovery task. Here we expand the (l,d)–motif problem to more accurately model this task and present a novel algorithm to solve this extended problem. This algorithm is guaranteed to find all (l,d)–motifs in a set of input sequences with unbounded support and length. We demonstrate the performance of the algorithm on publicly available datasets and show that the algorithm deterministically enumerates the optimal motifs.

Keywords: pattern discovery, motif, binding site, algorithm

1 Introduction

Four years ago, Pevzner and Sze [12] noted that despite significant advances in pattern discovery, there were still gaping holes in our ability to identify and enumerate frequent patterns in biological sequences. Experimental noise and error were not the only significant issues, as the community was still incapable of solving certain problems with purely synthetic data and no worry of experimental or gross error. One such problem, defined below, was the (l,d)–motif challenge problem; it exposed the fact that certain motifs, despite having a strong consensus and being rather unlikely to occur at random in independent and identically distributed (i.i.d.) sequences, are extremely hard for most motif discovery algorithms to locate. The reason that these motifs are hard to locate is that even though they may deviate very little from a consensus sequence, their pairwise deviation tends to be rather large. Other false pairwise similarities are thus extremely likely to occur at random elsewhere in the dataset, and this random noise obscures the true motif's signal. Pevzner and Sze [12] presented two algorithms that looked towards solving this problem; Buhler and Tompa [2] followed suit by presenting a more effective algorithm. However, the problem is still not completely solved per se; difficulties exist in obtaining the correctly refined motifs and instances even for this simplified model of biology. In addition, though existing algorithms move towards solving this simplified problem, they are not nearly as helpful in addressing the biological realities that computational biologists face.

The (l,d)–motif problem was designed to model a biological phenonmenon that could not be addressed by previous computational methods. To that end, the abstraction served its purpose well; however, there is a fundamental disconnect between the system that the challenge problem models and the biological realities scientists face on a daily basis. It is this disconnect that has kept the motif challenge problem merely a theoretical, academic exercise up until now. It is important to create a connection between the abstracted problems we strive to solve and the systems to which those solution algorithms could ultimately be applied. While the (l,d)–motif problem described below is justifiably inspired by biology, it deviates from research conditions in a few key ways that ought to be modified to more accurately reflect experimental conditions and prompt the creation of truly useful search tools. This paper aims to point out those differences and reframe the problem in a manner more accurately reflecting the biological system being modeled.

1.1 Background on Motif Discovery

Motif (or pattern) discovery encompasses a wide variety of methods that are used to identify recurrent trends in data. In the most general sense, these methods include various forms of clustering and discriminant analysis techniques [5, 16]. Motif discovery tools that are used in the analysis of amino acid and nucleotide sequences are special in that they treat sequential data — that is, data in which there is a natural ordering, such as a sequence of nucleotides in a ribosomal binding site [14].

Some of the more popular motif discovery tools used in sequence analysis include Pratt [9], Teiresias [13], MEME [1], Gibbs sampling [11], Consensus [7], and Block Maker [6]. Of these, Pratt and Teiresias use regular expressions to represent motifs, whereas the others use either position weight matricies (PWMs) or position specific scoring matrices (PSSMs).

Regular expressions — which are also known as regular grammars or type–I grammars — are simple rules specifying allowed arrangements within a sequence of characters [4, 10]. Most motif discovery tools that are based on regular expressions restrict themselves to finding only a small sub–class of regular expressions. Within that class of patterns, they are typically exhaustive.

In contrast, most tools based on PWMs or PSSMs are not exhaustive — that is, they do not find all possible PWMs/PSSMs in a set of sequences. Rather, they find a single, (hopefully) globally optimal motif. Unlike regular expressions, PWMs and PSSMs are probabilistic models of motifs. Because of this, they typically contain more information than regular expressions and are well–suited for modeling phenomena poorly represented by regular expressions, including the well–known example of *cis*–regulatory binding sites.

The automated discovery of *cis*–regulatory elements, which has been studied extensively in the bioinformatics community, was the motivating problem for the novel algorithm described here. We found that, given a large set of sequences, popular PWM/PSSM–based tools were unable to find many binding sites due to the tools' lack of exhaustiveness. Typically these tools require intelligent guidance by the user; while some types of guidance are easily given, like limitation of input sequences to regions likely to contain a motif, other types may be beyond the average user's reach. It is in these cases that PWM/PSSM–based methods are most likely to fail. We also found that the regular expression–based tools performed poorly simply because binding sites are best described by probabilistic models.

1.2 Structure

Our objective, then, is to discover DNA sequence motifs in a way that requires as little *a priori* knowledge as possible. The challenge problem presented in previous works is a simplified model of this more general problem. We contend that our algorithm is of significant use in solving the open (l,d)–motif problem in a deterministic but computationally tractable nature.

In what follows, we will give a formal statement of the motif discovery problem as it applies to our algorithm. We will then detail the algorithm and develop a few short applications.

2 The Expanded (l,d)–Motif Challenge Problem

The original (l,d)–motif problem [12] can be paraphrased as follows:

Within a set of random DNA sequences with i.i.d nucleotides, a parent motif of length l is embedded in each sequence in a random location. Each time the motif is embedded,

> it is mutated in d locations. The (l,d)–motif problem is to recover the locations of the embeddings, knowing only the parameters l and d and that each sequence contains exactly one instance of the motif.

At first, this seems to be a reasonable simplification of the phenomenon of binding sites and other functional sites in DNA. It is not uncommon to have some ancestral sequence from which each motif occurrence is some short evolutionary distance away. This model accurately captures the difference between instance–instance similarity and instance–ancestor similarity. That is, even though a motif instance may be a very short distance from its ancestor (say, four mutations out of fifteen bases), any two instances of the motif may be significantly different from each other (eight mutations out of fifteen bases). This low degree of instance–instance similarity can occur rather frequently in random i.i.d. nucleotide sequences, thus obscuring the true evolutionary relationship of the motif instances (the signal) with purely random relationships of background nucleotides (the noise) [2, 3].

As discussed by Buhler and Tompa [2], local search methods (such as the common ones mentioned before) using typical initialization strategies encounter an insurmountable amount of noise when searching for some sparse motifs described by the (l,d)–motif problem. We would ideally like to be able to recover such motifs, since they are expected to occur by chance in every sequence with rather low probability (approximately 10^{-7}) [2, 3].

In a more realistic scenario, a researcher may not know the size l of the motif *a priori*. Instead, it is more likely that she would know the evolutionary distance between motif instances, i.e. the rate of mutation d/l. It is also unrealistic to mutate the embedded motif *exactly* d times; rather, the researcher is more likely to be interested in motifs that are d or fewer mutations away from each other. That is, in a real–world senario, we would more likely have a reasonable estimate of the upper limit d/l of the mutation distance between embedded motifs. There may also be multiple, different motifs in the dataset. Finally, as experimental data are commonly rife with noise, it is likely that some of the sequences may be false–positive candidates for the motif; that is, some sequences may contain no motifs at all.

With these issues in mind, we define an extended (l,d)–motif problem as follows:

> Within a set of random DNA sequences with i.i.d. nucleotides, a parent motif of length $\geq L$ is embedded zero or more times in each sequence in a random location, such that the motif has been embedded a total of k times in the data set. Also, each time the motif is embedded it is mutated such that there are no more than d mutations over any window of l nucleotides (that is, the rate of mutation is d/l). This process is repeated for any number of parent motifs, each with the same l and d, but possibly different L. The extended (l,d)–motif problem is to recover the locations of the embeddings for every parent motif without any *a priori* knowledge of where they might be, but only knowing the parameters l and d.

We will refer to this formulation as the "extended" (l,d)–motif problem and the previous formulation as the "restricted" (l,d)–motif problem. In what follows, we detail an algorithm for solving both the extended and restricted (l,d)–motif problems.

We say that a motif, p, is just a data structure with two features: a width, $\mathscr{W}(p)$, and a list of locations in the data where the motif has been embedded, $\mathscr{L}(p)$. A motif has the property that the locations in $\mathscr{L}(p)$ are all within a Hamming distance of $2d$ from each other over every window of size l.

We say a maximal motif is a motif which has the following properties:

1. The width of the motif cannot be increased without producing a motif with fewer embeddings (i.e., without $|\mathscr{L}(p)|$ decreasing);

2. The location of all instances cannot be made earlier without producing a motif with fewer locations (i.e., without $|\mathscr{L}(p)|$ decreasing); and

3. No new embedding can be added to the location list without creating a motif of shorter width (i.e., without $\mathscr{W}(p)$ decreasing).

These criteria can be summarized qualitatively by stating that a maximal motif is not "missing" any embeddings and is as wide as possible, and thus hopefully as specific and sensitive as possible.

We will call the Hamming distance function $\mathscr{H}$, where $\mathscr{H}$ takes two windows of size l from our sequence set, and returns a real–valued number equal to the number of characters that differ between the two windows.

Using these definitions, we note that the extended (l,d)–motif problem can be solved another way. Given a sequence set and the function $\mathscr{H}$, finding all maximal (l,d)–motifs in the sequence set will identify all possible planted motifs and enumerate the locations of all instances of those motifs.

3 An Algorithm for the Extended (l,d)–Motif Problem

3.1 Scanning Phase

The input to our algorithm is a set of nucleotide sequences $S = \{s_1, s_2, \ldots, s_n\}$, where sequence i has length W_i. So, for example, the j^{th} member of the i^{th} sequence is denoted by $s_{i,j}$ and is one of the characters `A,T,G,C`.

Typically, a user is interested in motifs that have a minimal length (i.e., motifs of length 1 are not interesting, whereas motifs of length 10 might be depending on the context). We denote this user–supplied length as L and we define a symmetric matrix A of size $N = \sum_{i=1}^{n}(W_i - L + 1)$. That is, A is a matrix with one index for each window of size L in our entire sequence set. The 10^{th} window of size L in the 5^{th} sequence would be expressed as $s_{5,10:10+L-1}$, where "$10 : 10 + L - 1$" denotes "position 10 through position $10 + L - 1$, inclusive". To keep track of which window corresponds to each index in A, we define the function $\mathscr{M}(s_{i,j:j+L-1}) \mapsto q \in [1, N]$. (For the sake of simplicity in later use, we define $(s_{i,j} + 1)$ to be $s_{i,j+1}$, unless $s_{i,j+1}$ does not exist, in which case $(s_{i,j} + 1)$ is undefined.) Similarly, $\mathscr{M}^{-1}(q) \mapsto (s_{i,j:j+L-1})$ such that $i \in [1, n]$ and $j \in [1, W_i - L + 1]$.

Thus, $A_{i,j}$ is equal to $\mathscr{H}(\mathscr{M}^{-1}(i), \mathscr{M}^{-1}(j))$.

3.2 Convolution Phase

At this point we have a matrix A that contains the pairwise distances between each window in the dataset. One could also say that that A is simply a distance matrix that could be used to plot the windows in some high–dimensional space. In that sense, we are interested in clustering these windows together to form "elementary motifs".

We define a clustering function $\mathscr{C}(A) = c^0 = \{c_1^0, c_2^0, \ldots, c_Z^0\}$ where each c_i^0 is a set of indices in A and $c_i^0[q]$ is the q^{th} member of c_i^0. Note that $\mathscr{C}$ can be any function; this work uses a maximal clique–finding algorithm described elsewhere [15]. Also note that each c_i^0 may have a non–null intersection with any c_j^0. We call each c_i^0 an "elementary motif" at level 0 (or equivalently, of length L).

Next we define the "directed intersection" of two elementary motifs, $c_i^0 \curvearrowright c_j^0 = c_k^1$, where c_k^1 is the set of those indices in c_i^0 such that $\mathscr{M}(\mathscr{M}^{-1}(c_i^0[q]) + 1)$ is in c_j^0. That is, c_q^1 is the set of indices in c_i^0 that are located, in the sequences S, one position earlier than the indices in c_j^0. c_q^1 is then an elementary motif corresponding to a window of size $L + 1$.

We also define the operation "$\sqsubset$" as follows: $c_i^0 \curvearrowright c_j^0 \sqsubset c^1$ is true if the set of indices $c_i \curvearrowright c_j$ is a subset of the indices in any member of c^1. Basically, this a check of whether $c_i \curvearrowright c_j$ is redundant with a member of c^1.

Our objective is to find all the maximal motifs in the sequence set using the elementary patterns. We call this set of motifs P. Of course, there are various pruning strategies that can be implemented to cut down on computation time, limit memory use, and eliminate motif discovery, all without loss of generality. Details on these implementation specifics can be found elsewhere [8]. A broader overview will suffice for the purposes of this work, and so we present the following sketch of our brute–force algorithm that faithfully describes our method of finding members of P:

```
begin
  $n := 0$
  while $|c^n| \neq 0$ do
        for $i := 0$ to $|c^n|$ step 1 do
            ismaximal := true
            for $j := 0$ to $|c^n|$ step 1 do
                $f := c_i^n \curvearrowright c_j^n$
                if $|f| \neq 0$
                  if $f \sqsubset c^{n+1}$ = false
                    $c^{n+1} := c^{n+1} \cup f$
                  fi
                  if $|f| = |c_i^n|$
                    ismaximal := false
                  fi
                fi
            od
            if ismaximal = true
              $P := P \cup c_i^n$
            fi
        od
        $n := n + 1$
  od
end
```

4 Solving the Restricted (l,d)–Motif Problem

The input set for the (l,d)–motif problem is any arbitrary set of n sequences, each with length W_i nucleotides. Most bioinformatics literature treatments use $W_i = 600$ and $n = 20$. Different versions of this problem have been discussed at length; the most commonly discussed is the (15,4) problem, while the (14,4) and other associated, more difficult problems are also addressed in the literature.

It has been shown before that the most commonly used motif discovery algorithms, including CONSENSUS [7], Gibbs sampling [11], and MEME [1], are unable to solve the restricted (15,4) problem. Algorithms that are capable of solving the restricted (15,4) problem have been presented in the literature. While some of these, including Winnower and SP-STAR [12], are unable to solve the more complicated (14,4) problems, others are able to address this and other, more difficult, problems with some degree of accuracy. These latter algorithms usually leave the deterministic realm, though, and rely on probabilistic methods to find the planted motifs.

On the other hand, our algorithm allows for exhaustive, deterministic solution of these problems. The (l,d)–motif problem solved by the above–mentioned tools is a degenerate case of the extended problem that our algorithm was designed to solve. Thus, our algorithm is not optimally tuned for solving the restricted (l,d)–motif problem in the least amount of time. Nonetheless, solving a range of the restricted (l,d)–motif problems is still a valuable check on the utility of our tool to make sure it can solve at least some of them in a reasonable amount of time. In addition, our exhaustive search allows for one to see how many other false signals are in the data. This can facilitate the assessment of statistical significance of results, certainly an important step in analyzing any proposed signal.

4.1 Solution Method

Our algorithm requires three user input parameters: l, g, and k. l is the minimum motif size and the size of the sliding window used for judging similarity between two sequences. g is the similarity threshold for any two windows to be deemed instances of the same motif; in this case, if two windows of length 10 are a Hamming distance of 2 away from each other, g would need to be 8 or less for the windows to be in the same motif. Finally, k is the support, or minimum number of motif occurrences required to report the motif to the user.

It is obvious that any two motifs of length l each being mutated d times from an ancestral sequence can differ at most at $2d$ locations. Thus, at least $(l - 2d)$ locations must be preserved in the motif. This observation lays the foundation for discovery of the hidden motifs. Our algorithm is run with parameters $l = 15$, $g = 7$, and $k = 20$ for the (15,4) problem. The discovery of the motif is then a straightforward combinatorial problem with deterministic discovery of the solution.

It is important to note, however, that our method will solve and return a superset of the restricted (l,d)–motif problem. That is, any group of d–mutants from a common ancestor can be described as having $(l - 2d)$ identical bases, but not all groups of sequences with $(l - 2d)$ identical bases can be used to synthesize an ancestor from which all group members deviate $\leq d$ bases. When there are a large number of "signal" motif members, there is usually sufficient overall deviation to prevent a $\geq d$–mutant from joining a motif group. However, at smaller support k, it is more likely to find motif instances that violate the d–mutant constraint. It is not desirable to immediately remove motifs with such members from the output, as they do still meet the constraints imposed by our parameter values; rather, we can use a simple post-processing method to note which motifs have readily obvious ancestors and thus are the most likely candidate signals.

4.2 Discussion

A few interesting observations can be made regarding the complexity of the algorithm and the quality of its solutions. First of all, the time to solution is not affected directly by the length of the motif to be discovered as in many other exhaustive methods. Rather, it is the sparseness or subtlety of the motif (or more accurately, the probability of the pairwise motif similarity occurring randomly) that has the most profound impact on the complexity of the algorithm. The most computationally expensive step is the clique-finding function, which increases in computation time with the number of edges (np–complexity at worst, though on average much better). For varying l and d, as two l-mers sampled randomly from the background are more likely to meet the threshold of similarity defined by l and d, there will be more false edges (similarities) in the graph, and thus the clustering algorithm will take longer. Motifs of widely different length may be (approximately) equally likely in the background distribution if d is set to a certain value for each. In this case, it would take almost exactly the same amount of time to find both motifs in the same input set. Of course, the size of the data set also has a significant impact on computation time, as for any algorithm; a larger input set causes more false occurrences of a potential motif, and the resulting distance matrix needs more time to be explored by our clique-finding algorithm.

Also, our method does not preclude discovery of more than one instance of a motif in any given sequence. Much like the re–framing of the (l,d)–motif problem presented above, this is more reflective of what one expects may happen in a real biological system: motifs of biological significance may occur more than once in a biosequence, and it behooves us to be able to discover all occurrences. In fact, in the original dataset for the (15,4)–motif problem used by Pevzner and Sze [12], there is actually an additional instance of the original motif that occurred completely by chance; this instance was discovered in our solution of the problem.

Finally, it is important to note the absolute accuracy of our results. In previous papers presenting algorithms to solve the (l,d)–motif problem, a metric called the performance coefficient is used to gauge the accuracy of the algorithms. This is defined as $\frac{K \cap P}{K \cup P}$, where K is the set of $l * s$ nucleotides representing the s motif instances each of length l and P is the set of $l * s$ nucleotides representing the s proposed motif instances of length l. Coefficients above .75 are usually deemed acceptable for these algorithms. Improved algorithms return results with coefficients of about 0.9 or 0.95. Examples of the performance of other algorithms are presented in Table 1. Clearly, our algorithm returns all coefficients of 1; that is, it will return the exact location of all motif occurrences. This is a notable improvement over other algorithms that may return approximate motif locations that then need to be verified and slightly adjusted or optimized by hand. In fact, in any given run of PROJECTION (the most accurate of the algorithms in Table 1), one will usually find that one or two (or even more) of the returned motif instances are not just imperfectly located, but are false positives.

Table 1: Performance on a range of (l,d)–motif problems with synthetic data. Data from other algorithms are from Buhler and Tompa [3]. GibbsDNA, WINNOWER, and SP-STAR are averaged over eight random instances, while PROJECTION is averaged over 100 random instances. Computation times for our proposed algorithm are averaged over three random instances.

l	d	GibbsDNA	WINNOWER	SP-STAR	PROJECTION	Proposed algorithm	Time
10	2	0.20	0.78	0.56	0.80	1.00	8 min
11	2	0.68	0.90	0.84	0.94	1.00	$<$ 1 min
12	3	0.03	0.75	0.33	0.77	1.00	10.5 h
13	3	0.60	0.92	0.92	0.94	1.00	10 min
14	4	0.02	0.02	0.20	0.71	1.00	$>$ 3 months
15	4	0.19	0.92	0.73	0.93	1.00	6 h
17	5	0.28	0.03	0.69	0.93	1.00	3 weeks

The computation time of our tool becomes unacceptable as the motifs become degraded beyond the (15,4) problem. This is to be expected for a deterministic algorithm as the probability of the signal reaches a level that causes many pairwise similarities to occur by chance. Since our strategy is generalized and exhaustive, we expect the computation times to be suboptimal. Beyond this table, one would benefit from other probabilistic or heuristic algorithms in order to solve the more difficult (l,d)–motif problems in an acceptable period of time. Fortunately, it seems to not be a too frequent occurrence to search for a (18,6)–motif in each of 20 biological sequences, so our algorithm should be of significant utility for common applications.

5 Solving the Extended Problem

Of course, in a real biological problem, one does not have nearly the same certainty in the contents of each biosequence as is allowed by the (l,d)–motif problem. This becomes evident upon analyzing the situations that the (l,d)–motif problem is meant to analyze, the most salient of which being the discovery of transcription factor binding sites. In order to come up with the candidate coregulated sequences, the results of laboratory experiments are analyzed to find which genes are sufficiently coexpressed. However, much of this data is prone to noise. Some genes may not be coexpressed, though they may seem to be due to some experimental aberration. Of those that are actually coexpressed, they may or may not be coregulated by the same transcription factor; it is a distinct possibility (and quite frequently a reality) that genes appearing to be coexpressed are not bound by any common factor. The same analysis follows for other situations for which the (l,d)–motif problem is an otherwise reasonable approximation: experimental noise prevents certainty that all input sequences are truly.

Other methods meant to be robust enough to solve the restricted (l,d)–motif problem will lose significant advantage in this more realistic, extended set of circumstances. Our algorithm was designed specifically to deal with the issues addressed by the extended challenge problem. It discovers, in a provably exhaustive and deterministic fashion, all motifs described in the extended problem definition. Other algorithms discussed previously in this paper are just not constructed to deal with such uncertainty in motif characteristics; as such, there is little way to accurately compare the performance of ours and other algorithms on the fully extended problem. Thus, it seems intuitive to simplify the extended problem to something more complicated than the restricted (l,d)–motif problem, but for which there is still a useful metric for comparison between ours and other algorithms. What follows are two cases (discussed qualitatively) which demonstrate the specific benefits of our tool for pattern discovery on (15,4) problems beyond the restricted version.

5.1 Case 1: An Underestimated Number of Motif Instances

One source of difficulty in the extended problem may be the uncertainty as to the exact number of motif instances. For this case, we still restrict ourselves to windows of size l with d mutations from a consensus sequence. However, we allow for uncertainty in the number of motif instances. For this case study, we instruct algorithms to find motifs with instances in at least 15 sequences when in fact there is an instance in every sequence. If an algorithm such as WINNOWER were to search for cliques across 15 sequences when in fact all 20 sequences had a motif instance, it would have a final graph with much more than the single signal that it usually hopes to obtain. PROJECTION's attempts to find 15 instances when 20 actually occur are similarly problem-ridden, returning different candidate motifs on different runs. These results would sometimes have significant overlap with initial planted motif, though at other times would have very little overlap. Most disturbingly, all of these proposed motifs would have approximately the same score, thus making it difficult to discern a truly useful motif from one constructed from background noise. Our algorithm, on the other hand, returned the initially planted motif along with other smaller patterns that still met the criteria for classification as a motif.

5.2 Case 2: Zero-or-One Motif Instances

In this next case, we analyze the impact of there being zero or one motif instances in each sequence. To implement this simplification, we instruct each algorithm to find the exactly 15 motif instances that are implanted across 20 sequences. This makes the problem astonishingly similar to the (l,d)–motif problem, with the exception that not every sequence contains a motif instance. This problem setup is thus significantly more realistic, as one does not expect every sequence to have a motif occurrence in every pattern discovery problem. Of course, this is still a simplification of reality, as one would not expect to know the exact number of motif instances. However, not even this gross simplification can salvage the efficacy of existing algorithms for the discovery of such subtle motifs. A study using PROJECTION found results that rarely approached acceptable levels and more frequently approached performance coefficients expected from purely random guessing. Again, though, our algorithm solved the problem with only a small increase in computation time over solving the original (l,d)–motif problem.

6 Conclusions

The benefit of our proposed algorithm is then obvious: deterministic and provably complete output even in the face of uncertainty in motif characteristics. The motifs could have been longer than 15 bases, could have had fewer mutations, or could have occurred in a variable number of sequences, and our tool would have found them. Its only obvious negative aspect is its computational expense. The restricted (15,4) problem took 6 hours, while the extended problem took 13 hours. Compared to the runtimes of algorithms like PROJECTION, which can be as low as five minutes for the restricted problem, these runtimes may seem extremely large. In practice, however, this computation time is far from unacceptable; one would not expect to often encounter the need to run motif discovery many times sequentially, particularly if the results being returned to the user are deterministically correct.

Perhaps even more importantly, we have reframed the challenge problem statement in a way that is more biologically meaningful; hopefully this new challenge will inspire other methods that outperform ours in some way. While a deterministic and exhaustive method is always welcome, for some problems it seems that a heuristic approach may provide a good balance between time and accuracy; we look forward to seeing new tools that address our amended problem with sufficient accuracy.

References

[1] Bailey, T.L. and Elkan, C., Fitting a mixture model by expectation maximization to discover motifs in biopolymers, *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2:28–36, 1994.

[2] Buhler, J. and Tompa, M., Finding motifs using random projections, *Proc. Fifth Annual Int. Conf. Comput. Biol.*, 69–76, 2001.

[3] Buhler, J. and Tompa, M., Finding motifs using random projections, *J. Comput. Biol.*, 9(2):225–242, 2002.

[4] Chomsky, N., Three models for the description of language, *IRE Transactions on Information Theory*, 2:113–124, 1956.

[5] Duda, R. and Hart, P., *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

[6] Henikoff, S., Henikoff, J.G., Alford, W.J., and Pietrokovski, S., Automated construction and graphical presentation of protein blocks from unaligned sequences, *Gene*, 163(2):GC17–GC26, 1995.

[7] Hertz, G.Z. and Stormo, G.D., Identifying DNA and protein patterns with statistically significant alignments of multiple sequences, *Bioinformatics*, 15(7-8):563–577, 1999.

[8] Jensen, K.L., Styczynski, M.P., Munsey, M., Rigoutsos, I., and Stephanopoulos, G.N., Gemoda: A generic motif discovery algorithm, in preparation.

[9] Jonassen, I., Collins, J.F., and Higgins, D.G., Finding flexible patterns in unaligned protein sequences, *Protein Sci.*, 4(8):1587–1595, 1995.

[10] Jurafsky, D. and Martin, J.H., *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, Upper Saddle River, New Jersey, 2000.

[11] Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., and Wootton, J.C., Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment, *Science*, 262(5131):208–214, 1993.

[12] Pevzner, P.A. and Sze, S.H., Combinatorial approaches to finding subtle signals in DNA sequences, *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 269–278, AAAI Press, 2000.

[13] Rigoutsos, I. and Floratos, A., Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm, *Bioinformatics*, 14:55–67, 1998.

[14] Rigoutsos, I., and Floratos, A., Parida, L., Gao, Y., and Platt, D., The emergence of pattern discovery techniques in computational biology, *Metabolic Engineering*, 2:159–177, 2000.

[15] Tomita, E., Tanaka, A., and Takahashi, H., *An Optimal Alorithm for Finding All the Cliques*, IPSJ technical report of SIG algorithms, 12:91–98, The University of Electro-Communications, 1989.

[16] Tou, J.T. and Gonzalez, R.C., *Pattern Recognition Principles*, Addison–Wesley, Reading, Massachusetts, 1974.