BIOINFORMATICS

Vol. 18 no. 10 2002
Pages 1374–1381

Finding motifs in the twilight zone

U. Keich* and P. A. Pevzner

Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093, USA

Received on February 1, 2002; revised on March 25, 2002; accepted on April 5, 2002

ABSTRACT

Motivation: Gene activity is often affected by binding transcription factors to short fragments in DNA sequences called motifs. Identification of subtle regulatory motifs in a DNA sequence is a difficult pattern recognition problem. In this paper we design a new motif finding algorithm that can detect very subtle motifs.

Results: We introduce the notion of a multiprofile and use it for finding subtle motifs in DNA sequences. Multiprofiles generalize the notion of a profile and allow one to detect subtle patterns that escape detection by the standard profiles. Our MULTIPROFILER algorithm outperforms other leading motif finding algorithms in a number of synthetic models. Moreover, it can be shown that in some previously studied motif models, MULTIPROFILER is capable of pushing the performance envelope to its theoretical limits.

Availability: http://www-cse.ucsd.edu/groups/bioinformatics/software.html

Contact: keich@cs.ucsd.edu

INTRODUCTION

In its simplest form, the motif finding problem can be formulated as follows: given a sample of sequences and an unknown pattern (motif) that is implanted at different unknown positions, can we find the unknown pattern? If an l-letter pattern is implanted without mutations in each sequence, one can find the motif by a straightforward frequency analysis of all l-letter substrings that appear in the sample. However, biological motifs are subject to mutations and usually do not appear exactly. A natural model is to allow the unknown pattern to be implanted with some mutations (e.g. mismatches) in the sample sequences. An (l, k)-motif (or pattern) is a pattern of length l that is implanted in sequences from the sample with at most k mutations (the implants are called *instances*).

Motif finding resulted in a number of excellent software tools based on greedy algorithms (CONSENSUS—Hertz and Stormo (1999)), Gibbs sampling (GibbsDNA—Lawrence *et al.* (1993)), EM algorithms (MEME—Bailey and Elkan (1995)), and other approaches. Pevzner and Sze (2000) asked: 'What are the limits of these motif finding algorithms?' They demonstrated that CONSENSUS, GibbsDNA, and MEME are unable, for the lack of a good starting point, to detect rather strong motifs, for example a $(15, 4)^*$-motif in the *Motif Challenge Problem* (one instance of a motif of length 15 with exactly four mutations is implanted in each sequence in a sample of twenty 600 bp sequences). These studies led to an algorithm (WINNOWER) that solves the Motif Challenge Problem. Recently, Buhler and Tompa (2001) were able to find more subtle motifs that the Pevzner–Sze algorithm failed to detect. They accomplished this through a novel approach to providing an EM algorithm with a good starting point (seed).

Probably the best tools for finding consensus based motifs (Stormo, 2000) are the pattern-driven algorithms (Brazma *et al.*, 1998) that test all 4^l l-letter patterns, score each pattern by the number of approximate occurrences in the sample (or by a more involved function) and find the high-scoring patterns (Staden, 1989; Pesole *et al.*, 1992; Wolfertstetter *et al.*, 1996; van Helden *et al.*, 1998; Tompa, 1999). However, an exhaustive search through all 4^l l-letter patterns becomes impractical for large l, and Tompa (1999) raised the problem of extending this approach for longer patterns.

To avoid the, potentially forbidding, exhaustive search through the set of all 4^l patterns, many algorithms generate a smaller set of *seed patterns* whose neighborhoods are then further explored via local search. For example, the *sample-driven* approach limits the set of seeds to patterns appearing in the sample (Bailey and Elkan, 1995; Fraenkel *et al.*, 1995; Rigoutsos and Floratos, 1998; Li *et al.*, 1999; Gelfand *et al.*, 2000; Pevzner and Sze, 2000). If, by chance, the pattern has an exact occurrence in the sample, this approach is as good as the pattern-driven approach. If not (which is usually the case for biological samples), the hope is that the implanted instances of the pattern appearing in the sample will reveal the pattern itself via local improvements. However, in the case of subtle signals, this approach needs to be taken with caution. The problem is that it is difficult to distinguish the instances of the pattern from many random substrings

*To whom correspondence should be addressed.

that are similar to these instances just by chance. Pevzner and Sze (2000) addressed this problem by developing the WINNOWER and SP-STAR algorithms and further testing them with a variety of samples generated in bacterial comparative genomics studies (Sze *et al.*, 2002). Although these algorithms worked for the challenge problem—that CONSENSUS, GibbsDNA and MEME failed to find—they did not work in the twilight zone of $(15, 4)^*$-motifs implanted in 1600 bp sequences.

In this paper we describe a new approach that finds very subtle patterns in the twilight zone and works very fast in practice. In particular, according to Pevzner and Sze (2000), CONSENSUS, GibbsDNA and MEME have all failed to find a $(15, 4)^*$-motif implanted in $N = 600$-letter sequences, SP-STAR failed at $N = 1000$, and WINNOWER became very time-consuming and unusable for $N > 1300$. While definitely improving on these algorithms, Buhler (2001) report they succeed in 16 out of 20 times in detecting the same $(15, 4)^*$-motif in twenty 2000 nucleotide sequences. They mention that they cannot attribute the failures to spurious, or strong random, motifs. Using Buhler and Tompa's scoring method we are able to successfully detect the same motifs more than 99% of the time (in about 1 hr and 15 min on a 500 MHz G4). Moreover, we can detect more than 98% of such motifs implanted in twenty 3000-long sequences.

Because the pattern-driven approach is too time-consuming and the sample-driven approach often misses subtle patterns, a natural idea is to design a hybrid approach that extends the search capabilities of the sample-driven approach while still avoiding the computational complexity of the pattern-driven approach. Two patterns are called *α-neighbors* if they differ by at most α substitutions. In an *extended sample-driven* (ESD) approach we look for (l, k)-motifs by generating all k-neighbors for every substring of length l in the sample (Waterman *et al.*, 1984; Galas *et al.*, 1985; Sagot *et al.*, 1995; Sagot, 1998). The number of patterns explored in this approach is roughly $nN\binom{l}{k}3^k$ versus 4^l in the pattern-driven approach, where n is the number of sequences of length N in the sample.

Although the ESD approach is faster than the pattern-driven approach for typical l and k, it is still slow in practice. Our new motif-finding algorithm is a fast emulation of the ESD approach. It is based on the observation that very few of the neighbors generated in the ESD approach may steer us to the pattern P while others are not likely to help in finding the real pattern. The question is: 'Can we still capture the pattern by generating very few neighbors instead of $\binom{l}{k}3^k$ neighbors for every substring in the sample?' Our new MULTIPROFILER algorithm achieves this goal both in the context of very subtle signals implanted in synthetic data as well as in looking for regulatory patterns, including ones which include spacers (positions with arbitrary nucleotides), in biological samples. In this paper we present these results and describe our algorithm. In a companion paper (Keich and Pevzner, 2002) we provide a detailed analysis of the performance of MULTIPROFILER in a proposed general framework of analysis of subtle motif finders.

NEW IDEAS

In this section we introduce the two new ideas behind MULTIPROFILER. The first one is the utilization of a neighborhood of each word in the sample (i.e. all neighbors of this word that appear in the sample) as a possible 'dictionary' that suggests the correct 'spelling' of that word. The second is the usage of multi-positional profiles.

Our goal is to find the pattern P which is the consensus back-bone of the motif. Let $I(P) = \{P_1, \ldots, P_n\}$ be the set of n instances of the motif. In a synthetic model the instances are implanted in the randomly generated sample according to some rule. For example, the motif can be an $(l, k)^*$ one, meaning that in each instance exactly k out of P's l positions are mutated. These instances can be implanted either in one long sequence $\mathcal{S}$, or, as in Pevzner and Sze's FM model, exactly one instance in each of the n sequences $\mathcal{S} = \{S_1 \ldots, S_n\}$. Alternatively, the motif can be an (l, ρ)-motif ($\rho < 1$), where the positions of each instance of P are mutated, independently of all other positions, at a constant rate ρ. Unless otherwise stated both P and $I(P)$ are unknown to us. Generally speaking, if the motif is very subtle, then fully recovering the set of instances, $I(P)$, is a hopeless task due to good random matches of P. Note, however, that this is a different issue than recovering P (Keich and Pevzner, 2002).

Assume for a moment that the set of instances, $I(P)$, is disclosed to us. Finding P in this case reduces to the trivial task of aligning $P_1, \ldots, P_n$ and deducing the consensus sequence from the positional profile. For example, if the instances we align are of a $(15, 4)^*$-motif, then the expected frequency of the correct nucleotide in each of the 15 positions equals to $\frac{11}{15} \approx 0.73$ while the expected frequencies of the other nucleotides are only about 0.09. Suppose now that we are not as fortunate and only P_1 is pointed out to us. Typically, P_1 is not a perfect image of P; indeed, in the FM model $d(P_1, P) = k$, where $d(V, W)$ is the Hamming distance between the l-letter words V and W (i.e. the number of mismatches between them). We would like to correct the mutated, or misspelled, positions of P_1, only we do not know who they are nor do we know the correct spelling of those positions.

By an α-neighborhood of P_1 in $\mathcal{S}$ we mean the set of all words in $\mathcal{S}$ which agree with P_1 except on at most α positions:

$$N_\alpha(P_1, \mathcal{S}) = \{W \in \mathcal{S} : d(W, P_1) \leq \alpha\}.$$

As explained next, this α-neighborhood of P_1 will be our reference dictionary as we attempt to correct the spelling errors in P_1 (or equivalently, to find P). For example, if by some miracle, $N_\alpha(P_1, \mathcal{S}) = I(P)$ then the errors in P_1 will be self evident from studying the *positional profile* (Gribskov *et al.*, 1987) of all the l-letter words in $N_\alpha(P_1, \mathcal{S})$. Indeed, this is the same trivial task we just alluded to above. Typically, however, $N_\alpha(P_1, \mathcal{S})$ would contain only a subset of $I(P)$ and, moreover, it would include many random words. Clearly, a delicate balance needs to be struck between allowing as many instances of P as possible into $N_\alpha(P_1, \mathcal{S})$ (read, large α) and decreasing the noise, or the random words in $N_\alpha(P_1, \mathcal{S})$ (read, small α).[†] For a given α, $N_\alpha(P_1, \mathcal{S})$ contains, say, m random words and, say, n_1 out of the n instances of P (see Table 1 for an example). In a typical application $m \gg n_1$ so there is a substantial amount of 'noise' in the alignment making it harder to detect P. We enlist bipositional and multi-positional profiles to help us separate the noise from the pattern. A *positional* profile of an alignment of words of length l, is a $4 \times l$ matrix $(q_{N,i})$ where $q_{N,i}$ is the frequency of nucleotide N in the ith position of the alignment.

A *bipositional* profile of the same alignment is a $16 \times \binom{l}{2}$ matrix $\{q_{N,M,i,j}\}$ where $q_{N,M,i,j}$ is the frequency of the dinucleotide NM at positions i, j $(i \leq j)$ respectively (see Table 1 for an example). Bipositional profiles are better in separating the pattern from the noise because the noise gets distributed among 16 dinucleotides instead of only four possible nucleotides, for the same noise, in positional profiles. For example, suppose we are given an alignment of words that contain five instances which perfectly conserve the first two letters of a pattern P=cc..., as well as 50 totally random words. If we try to recover the first two letters of the pattern using the positional profile of the alignment of all 55 words, then the probability that at least one of the 'c's will lose to the noise is about 0.50. This should be compared with the probability of only about 0.16 that the dinucleotide cc will lose to the noise in positions (1,2) of the bipositional profile.

Regardless of whether we use positional or bipositional profiles we have to address the issue of 'bias' when our alignment comes from the set $N_\alpha(P_1, \mathcal{S})$. Unlike the 'totally' random words mentioned in the previous paragraph, the random words that enter $N_\alpha(P_1, \mathcal{S})$ satisfy the very 'non-random' condition $d(P_1, W) \leq \alpha$. Thus, while the n_1 instances of P cluster around P, the m random words in $N_\alpha(P_1, \mathcal{S})$ cluster around P_1 and for sufficiently large m/n_1 the profiles would exhibit a bias toward P_1 (see for example the positional profile in Table 1). We handle this bias by ignoring all entries in the profiles that include the nucleotides of P_1 at the observed positions. For example, if P_1 starts with aa then we ignore any entry in the profile that has an a in one of the first two positions (see Table 1). Note that beside being forced upon us by the nature of the problem, this policy is consistent with our goal of correcting the misspelled positions of P_1. Although bipositional profiles now have only nine available dinucleotides over which the noise is distributed, positional profiles are now reduced to only three nucleotides, so there is still a clear advantage to using bipositional profiles as the next example demonstrates.

Suppose 20 instances of an $(8, 2)^*$-motif are implanted in a random sequence, $\mathcal{S}$, of 4000 nucleotides. Then, $N_4(P_1, \mathcal{S})$ will include all 20 instances in $I(P)$ and on the average about 450 random words W for which $d(W, P_1) \leq 4$. We can assume without loss of generality that P starts with cc and P_1 starts with aa. Then it is virtually certain that the most frequent nucleotide in the first two positions of the positional profile of $N_4(P_1, \mathcal{S})$ is a. We estimate that using positional profile we will fail to correct the two misspelled positions of P_1 about 43% of the time.[‡] Compare that with an estimated failure rate of only 15% when using bipositional profile for the same task.[§]

More generally, multi-positional profiles could gain us even more power to resolve the noise where appropriate. For example, consider 20 instances of a $(16, 4)^*$-motif implanted in a random sequence, $\mathcal{S}$, of 26 000 nucleotides. Then, $N_8(P_1, \mathcal{S})$ will include all 20 instances of P, and on the average about 700 random words W. Assuming that P starts with cccc and that P_1 starts with aaaa we are essentially guaranteed that a will dominate all four initial positions. We estimate that using positional profile to correct the misspelled positions of P_1 will fail us about 79% of the time, while when using quad-positional profile for the same task we will fail only about 9% of the time.

The above discussion was based on the assumptions that we were led to P_1 by the kindness of strangers. Deprived of that resource, we can try all possible options for P_1. For example, in the case of one instance per sequence as in the FM model, we can explore all the words of the first sequence, as one of them is bound to be P_1. We call such a candidate for P_1 a 'reference word'. For a given reference word A we proceed as outlined above: assuming we seek a $(15, 4)^*$-motif (exactly four mutations in each instance of the 15 letters pattern P), we look for possible

[†] As explained in the sequel for an $(l, k)^*$-motif, $\alpha = 2k$ is typically the sensible choice. Other motif models might require a more sophisticated analysis of the tradeoff between reliability and cost as outlined in the companion paper (Keich and Pevzner, 2002).

[‡] More precisely, using Monte Carlo simulations we find that [0.4297,0.4307] is a $1–2{\cdot}10^{-5}$ confidence interval for the probability that in at least one of the first two positions c would fail to be the most frequent letter among the nucleotides {c,g,t}.

[§] [0.1468,0.1476] is a $1–2{\cdot}10^{-5}$ confidence interval for the probability that cc would fail to achieve a majority among the dinucleotides which do not contain an a in the same two positions.

Table 1. Example of positional and bipositional profiles

$N_4(P_1, \mathcal{S})$	
P_1	aaCCCCCC
P_2	CgCtCCCC
P_3	CCaCCCaC
P_4	CCCgCCCg
W_1	aattccgc
W_2	agcaaccg
W_3	aactctaa
W_4	aggctacc
W_5	cacggcct

The positional profile

	1	2	3	4	5	6	7	8
a	5	4	1	1	1	1	2	1
c	4	2	6	3	6	7	6	5
g	-	3	1	2	1	-	1	2
t	-	-	1	3	1	1	-	1

The bipositional profile

	1,2	1,3...1,8	2,3...7,8
aa	3	- ... 1	- ... 1
ac	-	3 ... 3	3 ... 1
ca	1	1 ... -	1 ... -
ag	2	1 ... 1	- ... -
ga	-	- ... -	- ... -
at	-	1 ... -	1 ... -
ta	-	- ... -	- ... -
cc	2	3 ... 2	1 ... 3
cg	1	- ... 1	- ... 2
gc	-	- ... -	2 ... 1
ct	-	- ... 1	- ... 1
tc	-	- ... -	- ... -
gg	-	- ... -	1 ... -
gt	-	- ... -	- ... -
tg	-	- ... -	- ... -
tt	-	- ... -	- ... -

The $(8, 2)^*$-motif is based on P=cccccccc. There are four instances in the sample $\mathcal{S}$: $P_1, \ldots, P_4$. The neighborhood $N_4(P_1, \mathcal{S})$ also contains the random words $W_1, \ldots, W_5$ (some of which are at a substantial distance from the pattern P). As expected, due to the bias toward P_1 both the positional and the bipositional profiles suggest the pattern is P_1 rather than P. However, if we ignore the 'a's in the first two positions, as we would do when we try to correct the misspelled positions of P_1, the bipositional profile suggests the correct spelling of cc (excluding 'a's cc is top ranked for positions (1,2) of $N_4(P_1, \mathcal{S})$), while the positional profile erroneously suggests cg for the same two positions.

corrections to the misspelled positions of A (assuming it is indeed P_1) among the peaks of the quad-positional profile of $N_8(A, \mathcal{S})$. Note that as explained above these peaks should not contain any letter from A at the relevant positions. On the average the correct modification to P_1 is expected to appear in $(n-1)\binom{11}{4}/\binom{15}{4} \approx n/4$ of the P_is. This is significantly more than what we typically expect from a random 4-mer. Note that in restricting our attention to the peaks of the quad-positional profile, we risk losing the pattern P but as we demonstrate below the tradeoff is more than acceptable. This description sweeps under the rug a few technical details which are jointly addressed in the next section and in the companion paper (Keich and Pevzner, 2002).

THE MULTIPROFILER ALGORITHM

This section describes the search strategy utilized by the MULTIPROFILER algorithm. Given words, W and B of the same length, define $d(W, B)$ as the Hamming distance between them. The distance between a word W, of length l, and a sequence S is: $d(W, S) = \min_{B \in S} d(W, B)$, where the minimum is taken over all words (substrings) of length l in S. Finally, the total distance between W and the sample $\mathcal{S} = \{S_1, \ldots, S_n\}$, or simply the *total distance* of W, is $d(W) = \sum_i d(W, S_i)$. Total distance is our 'default' choice of scoring function though others can be used.

We use the term (k-)*wordlet* to denote a typically non-consecutive, k-letter subsequence of a word. A k-wordlet is defined in terms of its k positions in a word and their content. For example, the word atcgaa contains the 3-wordlet -t--aa. Subsequences of wordlets are called *syllables*, so -t---a is a syllable of size 2 of the 3-wordlet -t--aa. Note that like wordlets, syllables are *not* consecutive in general. A syllable, or a wordlet, is *disjoint* from a word if it differs from the word in all its positions. For example, the wordlet -t--aa is disjoint from the word cccatt but not from the word tttttt.

For simplicity, we concentrate in this section on the fixed mutation (FM) model (Pevzner and Sze, 2000), i.e. one instance of the pattern P, with *exactly* k mutations is randomly implanted into each of the sequences in the sample $\mathcal{S} = \{S_1, \ldots, S_n\}$. We first describe the simplest variant of the algorithm. We choose one *reference sequence*, say S_1 and we sequentially consider each of the l-letter words in S_1 as a *reference word*, denoted by A. If A coincides with P_1, the instance of P in S_1, then it contains exactly k mutated positions. Let $\gamma(P_1)$ denote the wordlet which correctly modifies P_1 (see Table 2 for an example).

Our main idea is that some of the other P_is, should have either preserved $\gamma(P_1)$ in its entirety or at least some syllables of it. Of course, we do not know the locations of the other P_is, however, we do know that $d(P_1, P_i) \leq 2k$.

Table 2. Examples of modifying wordlets

P	agtttgacctgacgc
P_1	aTAttgacGtgaTgc
$\gamma(P_1)$	-gt-----c---c--

$\gamma(P_1)$, the correct modification of P_1.

A	cgcgtttaagagacc
γ	---A-----C---GT
A_γ	cgcAtttaaCagaGT

A wordlet γ modifies a reference word A, to produce the modified word A_γ.

Table 3. The parameters of MULTIPROFILER

l	motif length
k	wordlet size or number of misspelled positions
α	radius of the neighborhood around each reference word
s	syllable size
n_r	number of reference sequences
β	score computing threshold

Upon deciding on a motif model we set these parameters according to our resources subject to the tradeoff between maximizing the detection rate and minimizing the expected cost. As of writing this paper these setting are found by trial and error based on the calculation presented in the companion paper (Keich and Pevzner, 2002).

It follows that with $\alpha = 2k$, each P_i $(i = 2, \ldots, n)$ will reside in the corresponding list of A's α-neighbors: $\mathcal{J}_i = \{B \in S_i : d(B, A) \leq \alpha\}$, where B denotes a word of length l. Since we know that the P_is are present in $N_\alpha(P_1, \mathcal{S})$ $(= \cup_i \mathcal{J}_i)$, and that $\gamma(P_1)$ will, most likely, appear in a few of those P_is, all we need to do is to scan the (multi) k-positional profile of $N_\alpha(P_1, \mathcal{S})$ for k-wordlets that appear in at least β words from distinct sequences, where β is a predetermined threshold. These 'popular' wordlets which are disjoint from A[¶] are then used to appropriately modify the reference word (see Table 2 for an example), and then the total distance of the modified reference word, $d(A_\gamma)$, is found[‖]. The algorithm would therefore *detect* the motif, provided $\gamma(P_1)$ has been preserved in sufficiently many other P_is (for more on this definition of motif detection refer to the companion paper (Keich and Pevzner, 2002)).

As an aside we note that one can think of looking for 'popular' wordlets (which are disjoint from A) as, again, a motif finding problem, only this one has a different nature. First of all, this 'sub-motif' can be very weak, meaning that most likely there will be many random motifs (wordlets) that will be stronger than the correct wordlet. The reason our method works is that this motif lies in a much smaller search space, so we can apply an exhaustive search to detect these weaker signals and then test them back in the reference word.

Since it is more likely that parts of $\gamma(P_1)$ are preserved rather than the whole wordlet, we introduce a variant of the algorithm that looks for syllables of $\gamma(P_1)$ that are present in $N_\alpha(P_1, \mathcal{S})$. For example, if the syllable size is $s = 2$ and the best match to the wordlet γ=-a--c-cg--- in $\mathcal{J}_i$ is -a--t-cg---, then we say that $\binom{3}{2} = 3$ of γ's syllables are present in $\mathcal{J}_i$ and that the count of (syllables of) γ in the sequence S_i is 3: $C(\gamma, S_i) = 3$. The *count* of γ in the whole sample $\mathcal{S}$ is $C(\gamma) = \sum_{i=2}^{n} C(\gamma, S_i)$. As before, if $C(\gamma) \geq \beta$, $d(A_\gamma)$, the total distance of the reference word A modified by γ (see Table 2), is computed and ranked. If $A_\gamma = P$ then the motif is detected. Note that setting $s = k$ returns us to the first variant of the algorithm. In this case, $C(\gamma)$ is the count of the number of sequences S_i for which γ is present in $\mathcal{J}_i$.

In both variants of the algorithm described above the threshold β regulates the performance of the algorithm: a decrease in β will increase both the cost and the reliability of the algorithm. All other parameters of the algorithm being fixed, the only term in the overall cost analysis which varies with β is the expected number of random wordlets γ for which $C(\gamma) \geq \beta$, which is essentially the expected number of patterns whose score will be computed. There are, however, other non-negligible costs such as the cost of counting the wordlets/syllables. The detection rate is essentially given by $\mathrm{P}\left[C(\gamma(P_1)) \geq \beta\right]$, where $\gamma(P_1)$ is the correct modification of the mutated wordlet in P_1.

So far we used only the first sequence in the sample, S_1, as our reference sequence, but we can use any number of the sequences as a reference sequence. For example, using all n possible sequences for reference improves the detection rate (the probability that the hidden pattern is detected) considerably but increases the computational complexity. However, even with a naive implementation we do not have to incur the full n-fold increase in cost. The main advantage of using n reference sequences instead of one is that we can increase β so that the cost per reference sequence will decline, and although the detection rate per reference sequence will decline as well, the overall detection rate will increase significantly. Since in the case of our challenge problem, the costs of the syllable size $s = 4$ variant have a steeper gradient as a function of β (due to lower fixed costs), it is better suited to take advantage of the move to n sequences as can be verified by Figure 1.

We end this section with a reference to Table 3 which summarizes the parameters of the MULTIPROFILER.

[¶] It is important to realize that in order to overcome the bias toward P_1 in $N_\alpha(P_1, \mathcal{S})$ we only look for wordlets that are disjoint from A.

[‖] A considerable amount of time can be saved by noting that the relevant total distance can be computed directly from $N_\alpha(P_1, \mathcal{S})$.

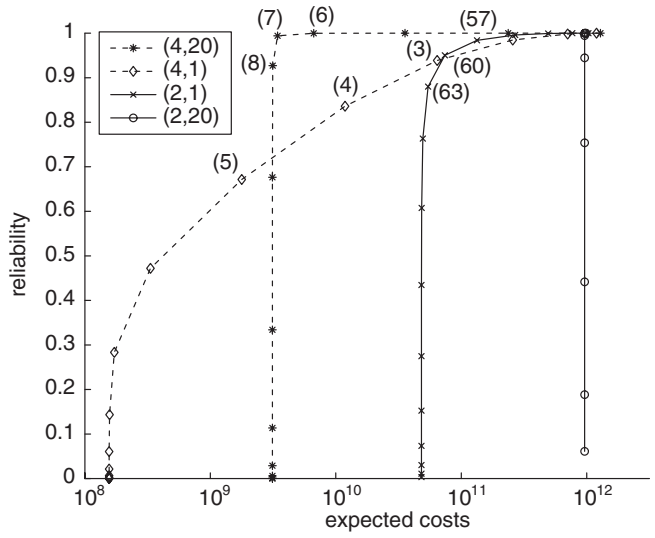

Fig. 1. Performance graphs of MULTIPROFILER The four expected costs versus reliability graphs above reflect the performance tradeoff of four 'variants' of MULTIPROFILER applied to our FM challenge problem (a $(15, 4)^*$-motif in twenty 1600 bp sequences). Each variant is designated by a pair (s, n_r), where s is the syllable size and n_r is the number of reference sequences used. Each point on the graphs corresponds to a different setting of β, the score computing threshold. A few values of β are provided (in parentheses) for orientation purpose. The vertices are joined by lines only to facilitate visualization. At high detection rates (above 93%), the (2,1) variant is more efficient than the (4,1): it costs more to obtain the same level of reliability using the (4,1) variant. Due to its high fixed costs the (2,20) variant is practically useless. Setting $\beta = 7$ with the (4,20) variant yields an overall detection rate of over 99.4% (less than 30% per sequence) at costs which translate to about 1.25 h on a 360 MHz G3. Achieving the same detection level using either the (4,1) variant ($\beta < 3$), or any of the $s = 2$ variants is visibly significantly more expensive. The vertical drop in each graph indicates the fixed costs associated with that variant. For an explanation on the derivation of the graphs refer to the companion paper (Keich and Pevzner, 2002).

BENCHMARKING

Pevzner and Sze (2000) establish that the well known weight matrix based algorithms, CONSENSUS, GibbsDNA and MEME fail to solve their challenge problem: finding $(15, 4)^*$-motifs implanted in 600 bp sequences. WINNOWER (Pevzner and Sze, 2000) is able to solve the challenge problem but fails to detect $(15, 4)^*$-motifs implanted in 1300 bp sequences. Buhler and Tompa's random projections algorithm, PROJECTION (Buhler and Tompa, 2001), was able to push the performance bar even further beyond 1300 bp.

In order to be able to compare MULTIPROFILER and PROJECTION we had to change our scoring function (total distance). The authors of PROJECTION chose to score a putative pattern P by counting the number of sequences in the sample whose distance to P is not bigger than k. Clearly, this scoring scheme heavily favors (l, k)-motifs whose instances appear in every sequence in the sample. As mentioned earlier, Buhler (2001) report they succeed in 16 out of 20 times in detecting a $(15, 4)^*$-motif in twenty nucleotide sequences of length 2000. They mention that the failures are not attributed to maximally scored random motifs, rather PROJECTION finds a suboptimal pattern. Testing our algorithm adapted to use PROJECTION's scoring scheme (instead of the total distance) we found that we are able to successfully detect the same motifs more than 99% of the time (in about 1hr and 15min on a 500 MHz G4). Moreover, we can detect more than 98% of such motifs implanted in twenty 3000-long sequences (about 3 hrs on a 500 MHz G4).[**] We also compared our results in the border line case of a $(9, 2)^*$-motif implanted in twenty 600-long sequences. As noted in Buhler and Tompa (2001) in four out of 20 cases PROJECTION failed to find a maximally scored pattern which of course has a score of 20. Our tests show that in less than 1 min running time, we essentially never fail to include the implanted motif among the patterns we list with a score of 20 (on the average there are 2.6 such—Buhler and Tompa (2001)).[††]

FINDING REGULATORY PATTERNS IN DNA SEQUENCES

We tested our algorithm on a few biological samples with known motifs. Our data consisted of:

- The sample of experimentally confirmed *E. coli* CRP binding sites containing eighteen 105-nucleotide sequences (Stormo and Hartzell III, 1989)
- Four samples from a variety of organisms taken from regions upstream of four eukaryotic genes: preproinsulin, dihydrofolate reductase (DHFR), metallothioneins, and c-*fos* (Blanchette (2001) and Buhler and Tompa (2001)).
- A collection of promoter regions from the yeast *S. cerevisiae* known to contain a shared promoter (Buhler and Tompa, 2001). The promoter regions were from genes SWI4, CLN3, CDC6, CDC46, and CDC47.

Table 4 summarizes the performance of our algorithm in these cases. To demonstrate the algorithm's flexibility, we

[**] These success rates are based on simulation studies of 250 000 random samples. In each simulated run we can efficiently determine whether or not MULTIPROFILER detects the implanted motif without resorting to executing the algorithm.

[††] Our algorithm succeeded to detect the implanted motif in all 100 000 simulation trials.

Table 4. Biological motifs detection

Sequence	Length	Found motif of length 20	Reference motif	Ref.
E. Coli CRP	1890	**TGTGA**nnnn**G**n**TCACA**nttt	**TGTGA**nnnn**G**n**TCACA**	(1)
Preproinsulin	7689	gc**AGACCCAGCA**ccagggaa	**AGACCCAGCA**	(2)
		aattgcag**CCTCAGCCCC**ca	**CCTCAGCCCC**	(2)
		g**CCCTAATGGGCCA**ggcggc	**CCCTAATGGGCCA**	(2)
DHFR	800	**TTCGCGCCAAACT**tgacngc	**TTCGCGCCAAACT**	(2)
Metallothionen	6823	ctc**TGCGCCCGG**ccccgtcc	**TTGCGCCCGG**	(2)
		ggt**GCTCTGCAC**c**C**ggcccg	**AGCTCTGCACTC**	(2)
c-*fos*	3695	**CCATATTAGG**AnATCTGcgt	**GGATGTCCATATTAGG**ACATCTG	(3)
Yeast ECB	5000	gtn**TT**t**CC**CgtTa**AGGAAA**a	**TT**t**CC**CnnTn**AGGAAA**	(3)

The first column provides the name of the gene. Length is the total number of base pairs in the sample. We only look for motifs of length $l = 20$. Under the reference column we provide either an established biological motif or one found by alternative algorithms. Positions for which no consensus letter was found (at least 50%) are reported as n. We do not report the new motifs found by our algorithm: (1) was taken from Stormo and Hartzell III (1989), (2) from Blanchette (2001), and (3) from Buhler and Tompa (2001).

set the length of the signal in all the samples to $l = 20$ even though this is not the case in any of the samples. In all cases any arbitrary choice of k worked and the runs took a few seconds each. For example, for the preproinsulin sample we set $l = 20$, $k = s = 4$, $n_r = n = 4$, $\alpha = 6$, $\beta = 2$ and the running time was 16 seconds on a single processor PowerMac.

DISCUSSION

Although the description of MULTI-PROFILER was somewhat specialized to the case of an $(l, k)^*$-motif, it should be noted that the algorithm can easily be modified to handle other motif models just as well. For example, by simply applying the previously described procedure over a range of possible wordlet sizes $i = 0 \ldots k$, we can handle an (l, k)-motif (where each instance has *at most* k mutations). Indeed, this version was applied to some of the biological samples studied in the previous section . The same approach will work for an (l, ρ)-motif where the positions of each instance are independently misspelled at a rate $\rho < 1$ (this is the motif model that is used in the VM model defined in Pevzner and Sze (2000)). We can also modify the algorithm so it could be applied in an iterative fashion which can be useful for $(l, k)^*$-motifs with relatively large k (work in progress).

Although the models that we use to study our algorithm call for exactly one instance of the signal, P_i, in each sequence, the existence of more than one instance (invaded samples), as well as none (corrupted samples), is tolerated by our algorithm. Indeed, the results presented in the previous section demonstrate that our algorithm is flexible and robust enough to identify biological motifs (even ones with spacers) in corrupted and biased samples without any modifications of the basic algorithm and that no exact prior knowledge of the pattern length is required. Admittedly, as was pointed out by a keen eyed referee, these biological examples are much less subtle than the synthetic models which this algorithm was originally designed to handle. 'In particular, they have all been found previously using standard profile-driven motif finders'.

Throughout the paper we assumed the background distribution is uniform. Heavily biased samples should require a slightly more sophisticated metric than the Hamming distance as well as more involved counting function. As for gapped patterns, one approach might be to look for spatial relations between discovered consecutive patterns. Significant improvement can be also be expected once a mechanism to weed out 'parasite signals' will be implemented (e.g. Blanchette and Sinha (2001)).

We end with addressing a couple of issues that might raise some concerns among experienced readers. The first is the issue of profiles or weight matrices versus consensus based patterns (Stormo, 2000). Although MULTIPROFILER adopts a combinatorial consensus approach to motif finding, it is also able to present its results in the form of profiles (similarly to MEME and others). The top scoring patterns can also be used as seeds for stochastic optimization algorithms. Moreover, since a typical profile of length l corresponds to a pattern of length l formed by the most frequent nucleotides in every position of the profile, a search with this consensus *pattern* would typically return the correct motif leading to a successful reconstruction of the original profile. Thus, for a typical profile, the pattern-driven approaches might be at least as good as the profile-based approaches for many biological samples. In fact, Sze *et al.* (2002), recently benchmarked different motif finding algorithms and demonstrated that pattern-based approaches may perform better than profile-based methods for some difficult biological samples. Similarly, Pevzner and Sze (2000) and Buhler and Tompa (2001) demonstrated that pattern-driven approaches perform better than profile-based methods on simulated samples with

implanted motifs. In summary, there is currently no evidence that profile-based methods perform better than pattern-based methods on either biological or simulated samples.

The second concern that the reader might have is that MULTIPROFILER was designed with synthetic motif models, such as the (l, k)-motif, in mind. The (l, k)-motifs and the (l, ρ)-motifs (where the positions of each instance are independently misspelled with probability $\rho < 1$) are both special cases of what we call 'diffused motifs'. These can be characterized by the fact that the misspelled positions in each instance are chosen *independently* of the other instances. As an anonymous observant referee points out 'this feature is a significant departure from the norm for published biological motifs, in which mutations occur preferentially on a small subset of positions. However, the assumption of independence makes the motif-finding problem harder rather than easier, so it should not be considered a flaw in the algorithm'.

ACKNOWLEDGEMENTS

We would like to thank Jeremy Buhler for providing us with the biological samples used for generating Table 4. We also extend our gratitude to anonymous referees for their numerous insightful comments.

REFERENCES

Bailey,T. and Elkan,C. (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, **21**, 51–80.

Blanchette,M. (2001) Algorithms for phylogenetic footprinting. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB-01)*. Montréal, Canada.

Blanchette,M. and Sinha,S. (2001) Separating real motifs from their artifacts. *Bioinformatics, Proceedings of the Ninth International Conference on Intelligent Systems for Molecular Biology*. Oxford University Press, Copenhagen, Denmark, pp. S30–S38.

Brazma,A., Jonassen,I., Eidhammer,I. and Gilbert,D. (1998) Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.*, **5**, 279–305.

Buhler,J. (2001) *Search Algorithms for Biosequences Using Random Projection*, Ph.D. thesis, University of Washington.

Buhler,J. and Tompa,M. (2001) Finding motifs using random projections. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB-01)*. ACM Press, Montreal, CA, pp. 69–76.

Fraenkel,Y., Mandel,Y., Friedberg,D. and Margalit,H. (1995) Identification of common motifs in unaligned DNA sequences: application to *Escherichia coli* Lrp regulon. *Comp. Appl. Biosci.*, **11**, 379–387.

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

Gelfand,M., Koonin,E. and Mironov,A. (2000) Prediction of transcription regulatory sites in Archaea by a comparative genomic approach. *Nucleic Acids Res.*, **28**, 695–705.

Gribskov,M., McLachlan,M. and Eisenberg,D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl Acad. Sci. USA*, **84**, 4355–4358.

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

Keich,U. and Pevzner,P. (2002) Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, **18**, 1382–1390.

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

Li,M., Ma,B. and Wang,L. (1999) Finding similar regions in many strings. In *Proceedings of the 31st ACM Annual Symposium on Theory of Computing*. Atlanta, Georgia, pp. 473–482.

Pesole,G., Prunella,N., Liuni,S., Attimonelli,M. and Saccone,C. (1992) WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences. *Nucleic Acids Res.*, **20**, 2871–2875.

Pevzner,P. and Sze,S. (2000) Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, San Diego, pp. 269–278.

Rigoutsos,I. and Floratos,A. (1998) Combinatorial pattern discovery in biological sequences. *Bioinformatics*, **14**, 55–67.

Sagot,M. (1998) Spelling approximate or repeated motifs using a suffix tree. *Lecture Notes in Computer Science*, **1380**, 111–127.

Sagot,M., Escalier,V., Viari,A. and Soldano,H. (1995) Searching for repeated words in a text allowing for mismatches and gaps. In *Proceedings Second South American Workshop on String Processing*. Valparaiso, Chile, pp. 87–100.

Staden,R. (1989) Methods for discovering novel motifs in nucleic acid seqences. *Comput. Appl. Biosci.*, **5**, 293–298.

Stormo,G. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, **16**, 16–23.

Stormo,G. and Hartzell III,G. (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl Acad. Sci. USA*, **86**, 1183–1187.

Sze,S., Gelfand,M. and Pevzner,P. (2002) Finding weak motifs in DNA sequences. In *Proceedings of the Pacific Symposium on Biocomputing 2002*. pp. 235–246.

Tompa,M. (1999) An exact method for finding short motifs in sequences with application to the Ribosome Binding Site problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Heidelberg, Germany, pp. 262–271.

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

Waterman,M., Arratia,R. and Galas,D. (1984) Pattern recognition in several sequences: consensus and alignment. *Bull. Math. Biol.*, **46**, 515–527.

Wolfertstetter,F., Frech,K., Herrmann,G. and Werner,T. (1996) Identification of functional elements in unaligned nucleic acid sequences. *Comput. Appl. Biosci.*, **12**, 71–80.