Finding motifs in the twilight zone

Uri Keich
Department of Computer Science and Engineering
University of California San Diego
La Jolla, CA 92093, USA
keich@cs.ucsd.edu

Pavel A. Pevzner
Department of Computer Science and Engineering
University of California San Diego
La Jolla, CA 92093, USA
ppevzner@cs.ucsd.edu

ABSTRACT

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 consensus sequences 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.

1. 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 enumeration 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), Gibbs sampling (GibbsDNA), EM algorithms (MEME), and other approaches. Pevzner and Sze, 2000 [14] 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* (instances of a motif of length 15 with 4 mutations are implanted once 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 [5] 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.

Probably the best tools for finding consensus based motifs are the pattern-driven algorithms (Brazma *et al.*, 1998 [3]) 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 [18]; Pesole *et al.*, 1992 [13]; Wolfertstetter *et al.*, 1996 [25]; van Helden *et al.*, 1998 [23]; Tompa, 1999 [22]). However, an exhaustive search through all 4^l l-letter patterns becomes impractical for large l and Tompa, 1999 [22] raised the problem of extending this approach for longer patterns. One way around this problem is to use the *sample-driven* approach that limits the search to the patterns appearing in the sequences from the sample (Bailey and Elkan, 1995 [1], Fraenkel *et al.*, 1995 [6], Rigoutsos and Floratos, 1998 [15], Li *et al.*, 1999 [12]; Gelfand *et al.*, 2000 [8], Pevzner and Sze, 2000 [14]). 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 that are similar to the pattern just by chance. Pevzner and Sze, 2000 [14] 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., 2001 [21]). Although these algorithms worked for the challenge problem – that CONSENSUS (Hertz and Stormo, 1999 [9]) , GibbsDNA (Lawrence et al., 1993 [11]) and MEME (Bailey and Elkan, 1995 [1]) failed to find – they did not work in the twilight zone of $(15, 4)^*$-motifs implanted in 1600 bp sequences. Below 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 [14], 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 and Tompa [4] 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 their scoring method we are able to successfully detect the same motifs in 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. Note that when looking for subtle motifs one is generally very likely to encounter random motifs whose score is at least as good as that of the sought motif (indeed that is the essence of our definition of a subtle motif given in Section 6). This raises the issue of how do we determine whether or not an implanted motif is detected by an algorithm (Section 6).

Since 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 still avoiding the computational complexity of the pattern-driven approach. Two patterns are called *k-neighbors* if they differ by at most k 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 [24] and Galas et al., 1985 [7]), and also Sagot et al., 1995 [17], and Sagot, 1998 [16]). 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, in biological samples.

We end the introduction by addressing a couple of issues that might raise some concerns among the experienced readers. The first is the issue of profiles or weight matrices vs. consensus based patterns ([19]). 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., 2001 [21], 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 [14] and Buhler and Tompa, 2001 [5] demonstrated that pattern-driven approaches perform better than the profile-based methods on simulated samples with implanted motifs. In summary, there is currently no evidence that the profile-based methods perform better than the 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".

2. NEW IDEA

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 text 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 (diffused) motif. Let $I(P) = \{P_1, \ldots, P_n\}$ be the set of 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 [14], 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 we are, of course, oblivious to both P and $I(P)$. Generally speaking, if the motif is very subtle recovering the exact set of instances, $I(P)$ is a hopeless task due to good random matches of P. Note that this is a different issue than recovering P though ([10]).

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 $d(W, W')$ is the Hamming distance between the words W 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 neither 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 obtained from aligning all the 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). In the case of an (l, k)-motif, $\alpha = 2k$ is usually the only sensible choice. Other motif models might require a more sophisticated analysis ([10]).

For a given α, $N_\alpha(P_1, \mathcal{S})$ contains m random words and n_1 ($\leq n$) 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 i-th 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 $i \leq j$ 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 4 possible nucleotides, for the same noise, in positional profiles. For example, suppose we are given an alignment of words that contain 5 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 loose to the noise is 0.50. This should be compare with the probability of only 0.16 that the dinucleotide `cc` will loose 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 9 available dinucleotides over which the noise is distributed, positional profiles are now reduced to only 3 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 4,000 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`. Moreover, we estimate that the probability that `c` would fail to be the most frequent letter among the remaining three (`c,g,t`) in at least one of the first two positions is about 0.43. Compare that with an estimated probability of 0.15[1] that `cc` would fail to achieve a majority among the dinucleotides (which do not contain an `a`) in the same two positions.

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 4 initial positions. We estimate the probability that `c` would fail to be the most frequent nucleotide among `c,g,t` in at least one of the first 4 positions at 0.79, while the probability that `cccc` would fail to be the most frequent in the same four positions (excluding 4-mers that include an `a`) is only 0.10[2].

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 4 mutations the 15 bp long P), we look for possible 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 loosing the pattern P but

[1] [0.4297,0.4307] and [0.1468,0.1476] are the respective $2 \cdot 10^{-5}$ confidence intervals

[2] [0.7873, 0.7881] and [0.0932, 0.0938] are the respective $2 \cdot 10^{-5}$ confidence intervals

Alignment of $N_4(P_1, \mathcal{S})$

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	-	-	...	-	-	...	-

Table 1: Example of positional and bipositional profiles.
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). Note that in the second column of the positional profile c is ranked third while the pair cc is top ranked for the mutated positions of P_1 (1,2) if we exclude all pairs that contain an a (all pairs above the dividing line).

as we demonstrate below the tradeoff is more than acceptable. This description sweeps under the rug a few technical details which would be addressed in the next section.

Although the hitherto discussion was somewhat specialized to the case of an $(l,k)^*$-motif, it should be noted that the algorithm can easily be modified so it can handle other motif models just as well. For example, by simply applying the previously described procedure over a range of $i = 0 \ldots k$, we can handle an (l,k)-motif (where each instance has *at most* k mutations). 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 [14]). 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).

3. 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 choice of a scoring function.

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.

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 2: Examples of modifying wordlets.

For simplicity, we concentrate in this section on the fixed mutation (FM) model (Pevzner and Sze, 2000 [14]) that assumes that 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 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 $d(P_1, P_i) \leq 2k$. It follows that with $\alpha = 2k$, each P_i $(i = 2, \ldots, n)$ will reside in the corresponding list of A's neighbors: $\mathcal{J}_i = \{B \in S_i : d(B,A) \leq \alpha\}$, where B is 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. It is important to realize that, we only look for wordlets that are disjoint from A.

One can think of looking for "popular" wordlets (which are disjoint from $\gamma(A)$) 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. That is, we use the popular wordlets to appropriately modify the reference word (see Table 2 for an example) and then find the total distance of the modified reference word[3]. The algorithm would therefore *detect* the signal ([10]) provided $\gamma(P_1)$ has been preserved in sufficiently many other P_is.

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 in the whole sample $\mathcal{S}$ is $C(\gamma) = \sum_{i=2}^{n} C(\gamma, S_i)$. As before, if $C(\gamma) \geq \beta$, the total distance of the reference word modified by γ, A_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 $\mathcal{J}_i$.

A detailed analysis of the performance of MULTIPROFILER can be found in [10], here we only report the highlights. In both variants of the algorithm described above the threshold β regulates the performance of the algorithm. On the one hand, if we set β too small we will be overwhelmed by random wordlets whose count will exceed β. On the other hand, if we set β too large our detection rate (the probability that the hidden pattern is detected) will drop. In order to make an informed decision about β we need to compute the expected number of random wordlets γ for which $C(\gamma) \geq \beta$. This expectation is the only term in the overall cost analysis which varies with β, though there are other non-negligible costs such as the cost of managing the multi-positional profiles, or equivalently, of counting the wordlets/syllables. We should then contrast the overall cost with the detection rate which 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. Tables 3 and 4 provide partial results in the case of a $(15, 4)^*$-motif (exactly 4 mutations in a pattern of length 15) in a sample of twenty 1500 bp sequences.

The tables demonstrate that, using one reference sequence, the $s = 2$ variant has an advantage over the $s = 4$ one at high detection rates. That is, we need to invest less effort in the $s = 2$ case to obtain the same or better signal detection rate with the $s = 4$ variant. However, it is important to note that counting pairs incurs a rather high fixed counting costs, thus increasing β speeds up the $s = 4$ variant much more significantly than it does to the $s = 2$ variant. This will be important next.

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

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

Table 3: Analysis of the MULTIPROFILER algorithm ($s = k = 4$).

β	expected cost of total distance computation	observed detection rate	observed detection rate using n reference sequences
3	$4.8106 \cdot 10^{10}$	0.9356	1.0000
4	$8.2245 \cdot 10^{9}$	0.8299	1.0000
5	$1.0718 \cdot 10^{9}$	0.6627	1.0000
6	$1.0964 \cdot 10^{8}$	0.4617	0.9998
7	$8.9781 \cdot 10^{6}$	0.2745	0.9933

β is the threshold for the total count of the wordlet. To find the expected total cost one should add the expected "fixed costs" (the ones which do not depend on β) of $1.37 \cdot 10^8 + 4.5 \cdot 10^6$ to the corresponding entry in the second column. The "observed detection rate" data was derived from a Monte Carlo simulation that generated 10^6 random samples of our problem. Note that if we set $\beta = 7$ and use n reference sequences then we get a detection rate of about 0.9933 at an expected cost of $20 \cdot (1.37 \cdot 10^8 + 4.5 \cdot 10^6 + 9.0 \cdot 10^6)$. Achieving the same detection level using either this variant with one reference sequence ($\beta > 3$), or the $s = 2$ variant (fixed costs of $20 \cdot 4.2 \cdot 10^{10}$), would be significantly more expensive. Note that all cost figures should be calibrated against the particular machine the algorithm is running on.

Table 4: Analysis of the MULTIPROFILER algorithm ($s = 2 < k = 4$).

β	expected cost of total distance computation	observed detection rate	observed detection rate using n reference sequences
50	$3.8971 \cdot 10^{11}$	0.9994	1.0000
55	$1.0393 \cdot 10^{11}$	0.9914	1.0000
56	$7.3879 \cdot 10^{10}$	0.9867	1.0000
57	$5.2098 \cdot 10^{10}$	0.9804	1.0000
58	$3.4310 \cdot 10^{10}$	0.9707	1.0000
59	$2.3319 \cdot 10^{10}$	0.9594	1.0000
60	$1.4789 \cdot 10^{10}$	0.9426	1.0000
65	$1.1853 \cdot 10^{9}$	0.7919	1.0000

The expected fixed costs is $4.2 \cdot 10^{10}$. This dominates the expected total distance cost starting at $\beta = 58$. Note that if we require detection rates over 90% and we use *only one* reference sequence, then this variant is better than the $s = k$ one. The "observed percentage" data was derived from a Monte Carlo simulation that generated 10^6 random samples of our problem.

n possible sequences for reference we improve our detection rate considerably but at an obvious increase in the 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 the costs, as a function of β, of the $s = 4$ variant have a steeper gradient, it is better suited to take advantage of the move to n sequences as can be verified by Tables 3 and 4.

4. BENCHMARKING

Pevzner and Sze, 2000 [14] establish that the well known weight matrices based algorithms , CONSENSUS, GibbsDNA and MEME fail to solve their challenge problem: finding $(15, 4)^*$-motifs implanted in 600 bp sequences. WINNOWER [14] 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 2001 [5], 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 and Tompa [4] 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 sub-optimal 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 in 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).[4] We also compared our results in the border line case of a $(9, 2)^*$-motif implanted in twenty 600-long sequences. As noted in [5] in 4 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 a minute running time, we essentially never fail to include the implanted motif among the patterns we list with a score of twenty (on the average there are 2.6 such [5]).[5]

5. FINDING REGULATORY PATTERNS IN DNA SEQUENCES

Although the FM model 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 making it a viable tool for finding a regulatory pattern. 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. These modifications are currently under development, however our described algorithm is flexible and robust enough to find biological motifs, used as tests in previous papers, without any modifications[6].

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 18 105-nucleotide sequences (Stormo and Hartzell, 1989 [20])
- 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 [2] and Buhler and Tompa, 2001 [5]).
- A collection of promoter regions from the yeast *S. cerevisiae* known to contain a shared promoter (Buhler and Tompa, 2001 [5]). The promoter regions were from genes SWI4, CLN3, CDC6, CDC46, and CDC47.

Table 5 summarizes the performance of our algorithm in these cases. To demonstrate the algorithm's flexibility, we 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. These tests demonstrate that our algorithm is able to identify signals (even ones with spacers) in corrupted and biased samples without any modifications of the basic algorithm and that no exact apriori knowledge of the pattern length is required. Significant improvement can be expected once a mechanism to weed out "parasite signals" will be implemented.

6. ANALYSIS OF SUBTLE MOTIFS AND MOTIF FINDERS

In this section we ask what is a subtle motif and how can we objectively evaluate the performance of a motif finder. We offer answers to these questions and demonstrate them in the context of a the previously mentioned FM and VM models, although much of the discussion in this section holds more generally. We assume that, relying on the scoring function, the motif finder tries to find a motif that is implanted in a sample randomly generated according to the *sample model*. The motif itself is generated and implanted in accordance with the *motif model*. We mainly discuss the total-distance scoring function, but in the context of the $(l, k)^*$-motif (FM model) we also consider the "sequence-count" scoring function used by Buhler and Tompa, 2001 [5], where we count

[4]These success rates are based on simulation studies of 250,000 random samples.

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

[6]As was pointed out by a keen eyed referee, these biological examples are much less subtle than the synthetic models studied here. "In particular, they have all been found previously using standard profile-driven motif finders".

Table 5: Biological motifs detection.

Sequence	length	Found motif of length 20	reference motif	Ref.
E. Coli CRP	1890	**TGTGAnnnnGnTCACAnttt**	**TGTGAnnnnGnTCACA**	(1)
preproinsulin	7689	**gcAGACCCAGCAccagggaa**	**AGACCCAGCA**	(2)
		aattgcagCCTCAGCCCCca	**CCTCAGCCCC**	(2)
		gCCCTAATGGGCCAggcggc	**CCCTAATGGGCCA**	(2)
DHFR	800	**TTCGCGCCAAACTtgacngc**	**TTCGCGCCAAACT**	(2)
metallothionen	6823	**ctcTGCGCCCGGccccgtcc**	**TTGCGCCCGG**	(2)
		ggtGCTCTGCACcCggcccg	**AGCTCTGCACTC**	(2)
c-*fos*	3695	**CCATATTAGGAnATCTGcgt**	**GGATGTCCATATTAGGACATCTG**	(3)
yeast ECB	5000	**gtnTTTCCCgtTaAGGAAAa**	**TTTCCCnnTnAGGAAA**	(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, 1989 [20], (2) from Blanchette, 2001 [2], and (3) from Buhler and Tompa, 2001 [5].

the number of sequences in the sample whose distance to the putative pattern is greater than k. The derivation of the numerical results presented in this section can be found in [10].

Loosely speaking, we consider a motif as subtle if its score is unremarkable when compared with the scores of some random motifs present in the same sample. This raises the question of what is the *twilight zone* of scores beyond which we can expect to start seeing random motifs. Clearly, this threshold depends on the sample model and on the choice of the motif scoring function. The sample model determines the distribution of the scores of random motifs and in particular it determines the expected number of random motifs at any given score. Generally, we expect the twilight zone threshold to lie around the score, for which the expected number of random motifs that exceed this score, is about 1 (clearly the choice of 1 is somewhat arbitrary, 2 or 1/2 are equally plausible however 10^6 or 10^{-6}, for example, are dubious choices). Figures 1 and 2 demonstrate how this threshold can vary with the the size of the sample and with the choice of the scoring function.

Assume for a moment that the score, γ, of the implanted motif is identical for all motifs generated according to our motif-sample model (for example, the sequence-count score will assign a score of 0 for any FM model). A plausible (though somewhat arbitrary) definition of a subtle motif is that its score lies beyond the twilight zone. In other words, we say the motif is *dim* if the expected number of random motifs with a score of γ or better is above 1. More generally though, the score of the implanted motif will vary not only with the particular motif, generated according to the motif model, but the score might also vary with the sample for the same particular motif (for example the total distance score will vary with good random matches of the pattern). Thus, in order to determine the "subtlety" of the motif we need some idea about how the score of the implanted motifs is distributed (see Figure 3 for examples of such distribution functions). We call a motif subtle or *dim* if its median score lies beyond the twilight zone. It is interesting to note the impact of the scoring function on this definition. For example, a $(15, 4)^*$-motif (FM model) implanted in 20 sequences of length 1600 is dim when viewed with the total

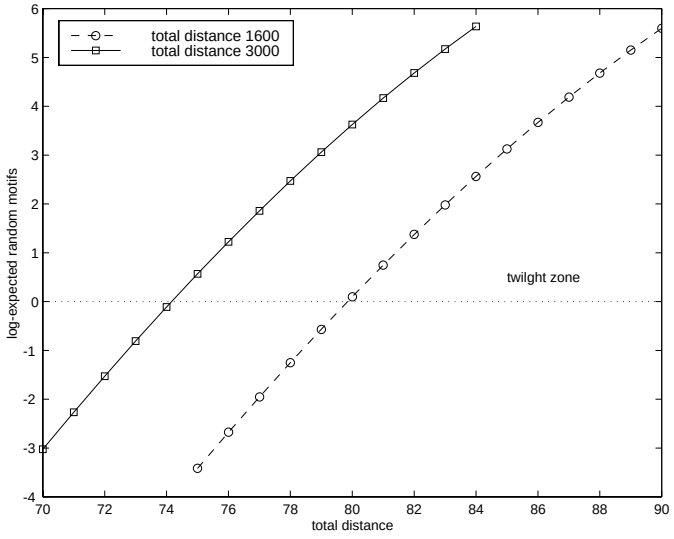

Figure 1: The expected number of random motifs as a function of the total distance score.
The point (x, y) is on the graph if y is the $\log_{10}$ of the expected number of random motifs with a score x or worse (i.e. x or bigger for the total distance score). The two graphs correspond to sample models of 20 sequences with 1600, respectively 3000 bp. The points are connected by lines merely to facilitate visualization. Note how the onset of the twilight zone shifts from a score of 80 to 74 when we move from 1600 to 3000 bp sequences.

distance score, however, the exact same motif "shines" quite brightly through the sequence-count binocular, indeed using this score we can extend the sequence length to over 3000 before this motif becomes dim (see [10] for details).

The performance of a motif finder is determined by its reliability (how well does it find motifs) and its complexity (at what cost). Clearly, how well an algorithm detects a motif might vary with the motif-sample model. However, it also depends on how one defines what constitutes a detected motif. Firstly, do we demand that all (or most) of the instances of the pattern be discovered, or are we willing to settle for the recovery of the pattern? Generally speaking, recover-

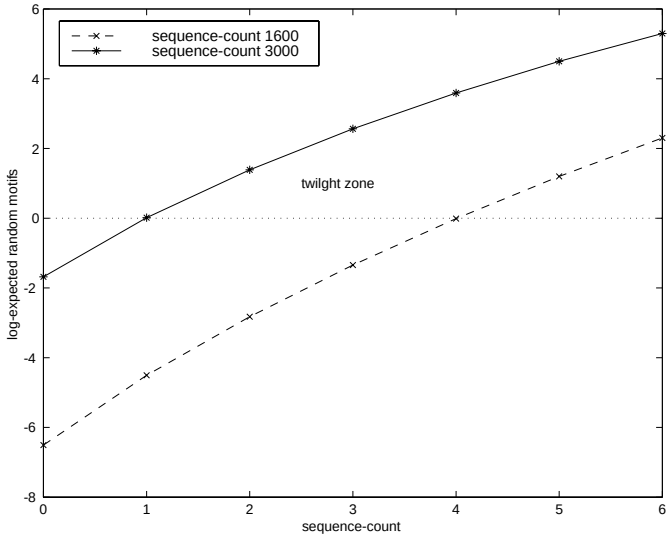

Figure 2: The expected number of random motifs as a function of the sequence count score.
In this example the onset of the twilight zone shifts from a score of 4 to 1 when we move from 1600 to 3000 bp sequences.

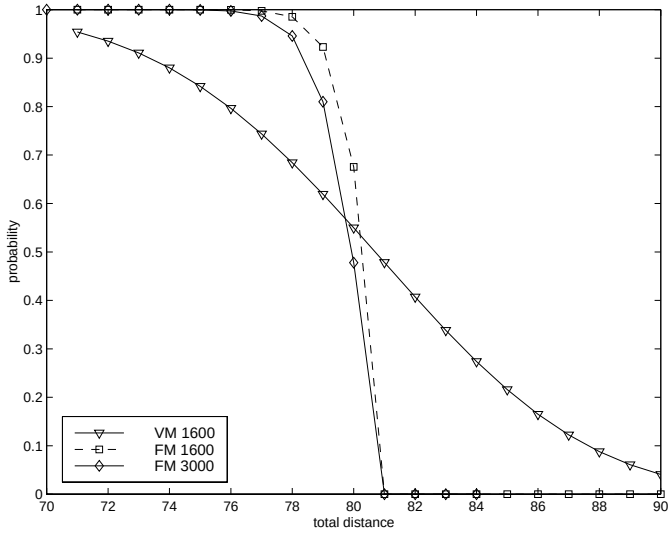

Figure 3: The distribution of the score of implanted motifs.
The point (x, y) is on the graph if y is the probability that the score of the implanted motif will be x or worse (i.e. x or bigger for the total distance). The points are connected by lines merely to facilitate visualization. The two FM models consists of a $(15, 4)^*$-motif implanted once in each of the 20 sequences with 1600 and 3000 bp. In the VM model the positions of the 15 bp pattern are mutated independently in each of the 20 instances (implanted in 1600 bp sequences) with probability 0.3.

ing the exact set of instances of a dim motif is a hopeless task due to good random matches of the pattern (for example, see the poor "average performance coefficient", or apc, reported in [4]). Secondly, since a subtle motif is unremarkable recognizing such a motif would often require some additional knowledge (obtained for example from biological experiments) to which the motif finders we consider here are not privy to. In particular by stating that "the implanted motif is found by the algorithm if it appears among the 5, 10, or 100 top scoring motifs reported by the algorithm", we unfairly place the burden of the motif subtlety on the shoulders of the algorithm: the algorithm might "fail" to detect the motif because its score is too low and/or the user was too impatient to look sufficiently down the ranked list of motifs suggested by the algorithm. To obtain a less user/subtlety dependent evaluation of the algorithm's reliability we say the unknown motif is *detected* if the algorithm correctly computes its score. For example, MULTIPROFILER [10] detects a (15,4) FM model implanted in 20 sequences of length 1600 over 99.4% of the time. One implication of this definition is that a dim motif is not necessarily one that is difficult to detect, for example, pattern-driven algorithms (Brazma *et al.*, 1998 [3]) that test all 4^l l-letter patterns are 100% reliable according to our definition and for a small l the cost is not too bad.

A more refined gauge of the subtlety of a motif can be obtained from what we call the *subtlety graph*. It is generated by combining the graph of the distribution of the motif scores with the graph of the (log-) expected number of random motifs per score. The subtlety graph provides us with the expected number of random motifs whose score will be as good as the given quantile of motif score. More precisely, the point (x, y) will be on the graph if 10^x is the expected number of random motifs whose score will be as good as the y-th quantile of motif scores, or equivalently, as good as any of the scores in the low $100 \cdot y$ percent of motif scores.

Turning this on its head we note this is equivalent to the following useful information: we will loose $100 \cdot y$ percent of the implanted motifs if we set a score based threshold that allows no more than 10^x random motifs (on average). An important advantage of subtlety graphs is that the information is presented in a way which is independent of the scoring function as you can verify with the examples in Figure 4.

7. ACKNOWLEDGMENTS

We would like to thank Jeremy Buhler for providing us with the biological samples used for generating table 5. We also extend our gratitude to anonymous referees for their insightful comments.

8. REFERENCES

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

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

[3] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5:279–305, 1998.

[4] J. Buhler. *Search Algorithms for Biosequences Using Random Projection*. PhD thesis, University of Washington, August 2001.

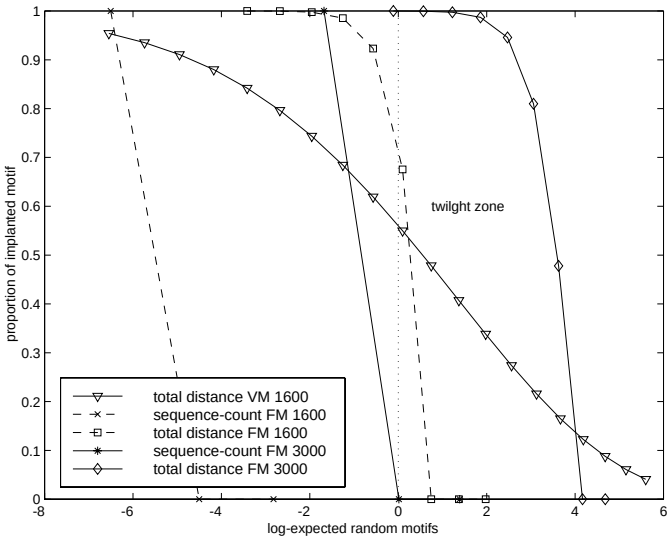

Figure 4: Subtlety graphs
The point (x, y) will be on the graph if 10^x is the expected number of random motifs whose score will be bigger than the y-th quantile of the implanted motif scores. The points are connected by lines merely to facilitate visualization. The points on the two sequence-count based graphs are located either at $y = 0$ or at $y = 1$ as the score of an FM implanted motif is constant at 0. Since the medians of all three total distance based examples are in the twilight zone, all three correspond to dim motifs. Note that the same motif implanted according to a 1600 bp FM model is dim under the total distance score but shines brightly with the sequence-count score. When the sequences are extended to 3000 bp, the motif is still not dim with the sequence-count score but is "practically" lost on the total distance score. Nevertheless, MULTIPROFILER is equally likely to detect this motif (over 98% of the time) using either score.

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

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

[7] D. Galas, M. Eggert, and M. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli. *Journal of Molecular Biology*, 186:117–128, 1985.

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

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

[10] U. Keich and P. Pevzner. Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, in press, 2002.

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

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

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

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

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

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

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

[18] R. Staden. Methods for discovering novel motifs in nucleic acid seqences. *Computer Applications in Biosciences*, 5:293–298, 1989.

[19] G. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.

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

[21] S. Sze, M. Gelfand, and P. Pevzner. Benchmarking of motif finding algorithms. In *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 235–246, 2002.

[22] M. Tompa. 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*, pages 262–271, Heidelberg, Germany, August 1999. AAAI Press.

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

[24] M. Waterman, R. Arratia, and D. Galas. Pattern recognition in several sequences: consensus and alignment. *Bulletin of Mathematical Biology*, 46:515–527, 1984.

[25] F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner. Identification of functional elements in unaligned nucleic acid sequences. *Computer Applications in Biosciences*, 12:71–80, 1996.