

Discriminative Motifs

Saurabh Sinha

Center for Studies in Physics and Biology

Box 25

The Rockefeller University

New York, NY 10021, U.S.A.

Ph: 1 212-327-8835, Fax: 1 212-327-8544

`saurabh@lonnrot.rockefeller.edu`

Abstract

This paper takes a new view of motif discovery, addressing a common problem in existing motif finders. A motif is treated as a feature of the input promoter regions that leads to a good classifier between these promoters and a set of background promoters. This perspective allows us to adapt existing methods of feature selection, a well studied topic in machine learning, to motif discovery. We develop a general algorithmic framework that can be specialized to work with a wide variety of motif models, including consensus models with degenerate symbols or mismatches, and composite motifs. A key feature of our algorithm is that it measures overrepresentation while maintaining information about the distribution of motif instances in individual promoters. The assessment of a motif's discriminative power is normalized against chance behaviour by a probabilistic analysis. We apply our framework to two popular motif models, and are able to detect several known binding sites in sets of co-regulated genes in yeast.

Keywords: Motif finding, TNoM score, Transcription Factor Binding Sites, Overrepresentation.

1 Introduction

Suppose we are given a set of DNA sequences that are hypothesized to contain several instances of a short pattern. How do we find that unknown pattern? This is the generic motif finding problem and it has been studied extensively, in different forms. One of its incarnations is the problem of finding transcription factor binding sites in the *promoter* regions of a set of co-regulated genes. Solving this problem provides a hypothesis about the interaction of genes with specific transcription factors, and such a hypothesis, when verified, leads to a better understanding of the regulation of gene expression. Binding site sequences are often approximately conserved across promoter regions, and the term “motif” means the common pattern in different binding sites of a transcription factor. Some previous approaches to *de novo* motif finding include those in (Lawrence et al., 1993), (Roth et al., 1998), (Bailey and Elkan, 1995), (Hertz and Stormo, 1995), (van Helden et al., 1998), (Tompa, 1999) and (Sinha and Tompa, 2000). Most approaches evaluate a motif by measuring how the frequency of its occurrences (allowing some variation) in the given promoter regions compares with the frequency in typical promoter regions. Our work belongs to this genre of overrepresentation based motif finding. The salient features of this work are:

- It takes a new view of motif discovery, treating it as a feature selection problem.
- It describes a general algorithmic framework that can be specialized to work with a large class of motif models.
- It utilizes information about the distribution of motif instances among the given promoters when assessing the motif’s overrepresentation, rather than looking only at the total count of occurrences.

The latter three of the motif finders mentioned above explicitly count the occurrences of a motif and compare this count against what is expected if the sequences had been generated by some suitable random process. A major strength of this approach is that evaluation of the overrepresentation of a motif has a sound statistical basis. A shortcoming of the technique is that it does not differentiate between the case when occurrences are distributed across the different promoters and the case when they are more concentrated in one promoter than in others, as long as the total count is the same. Given a set of genes believed to be co-regulated, it is typical to find binding sites in most of the promoters. In such situations, where we are interested only in motifs that are distributed over all or most of the promoters, the count based approach can lead to false positives. This is a common scenario and may arise for instance when there are several copies of a short repeat in one of the promoters.¹ Our work addresses this problem by proposing a new method to evaluate motif overrepresentation that utilizes information about the distribution of motif counts in individual promoters.

¹For instance, the motif AGTANNNNACAS occurs 10 times in 7 genes that are believed to be co-regulated by Daf-19p, in *C.elegans*. A count based approach considers this as overrepresented, even though all its 10 occurrences are in one promoter, and it is a short repeat rather than a binding site.

An overrepresented motif is, in a sense, one whose occurrences yield high *discrimination* between promoters that are believed to contain the binding sites (called “positive” promoters) and those that are not (called “negative” promoters). This suggests that a motif is a *feature* of the positive sequences that leads to a good classification of positive and negative promoters. We call such motifs “discriminative motifs” or “d-motifs” for brevity. Finding good d-motifs thus reduces to doing feature selection, a well studied topic in machine learning. Given a motif, we count its occurrences in each promoter, and measure the error made by the best linear classifier that classifies the positive and negative promoters based on these counts. The best d-motif is the one with the *most surprisingly low* classification error. The surprise is measured based on the *a priori* probability distribution of the count of the motif. Since it is difficult in practice to know negative promoters, we use randomly generated sequences as negatives. This heuristic is shown to work well in practice.

The notion of a d-motif is not restricted to any particular type of motif (e.g., consensus models with mismatches or degenerate symbols, or composite motif models), as long as there is a way to count occurrences of any motif of that type in a given promoter. The algorithm to find d-motifs is developed with minimal assumptions about the motif model, and can be applied to look for different types of motifs. Section 2 describes how a d-motif is evaluated, and Section 3 gives a general algorithm to find the best d-motifs. We demonstrate two applications of d-motif finding, each with a different motif model, in Sections 4 and 5. When run on real sets of co-regulated genes, the first application finds *simple* motifs that are verified as being known binding sites. (A simple motif models the binding site of a single transcription factor.) In the second application, the same technique is able to find a *composite* motif, which is a pair of simple motifs separated by a variable but restricted distance. (A composite motif corresponds to a pair of transcription factors, and has more variability than simple motifs.) Section 6 reports the results of experiments with synthetic data that evaluate the different aspects of the algorithm. In Section 7, we explain some of the choices made in the design of the d-motif framework.

1.1 Comparison with Previous Work

The work of (Hu et al., 2000) is similar to this work in some of the goals and methods. Their algorithm finds simple motifs that are well conserved, overrepresented and also distributed across promoters. This is done by computing a separate score for each of these three motif characteristics, and combining normalized versions of these three scores by taking their harmonic mean. This is a somewhat *ad hoc* method of combining the different axes of information, and we propose a more natural method with statistical justification. Their paper also attempts to classify two sets of promoters using a decision tree classifier that looks at occurrences of composite motifs. There is a feature selection step in this classification procedure that is equivalent to finding discriminative motifs. However, the feature selection is done based on classification score, which is not a powerful criterion for motif finding, especially when searching a very large feature (motif) space. As we see later, some motifs are expected to have a better classification score than others just by chance. Our

algorithm normalizes raw classification scores by computing their p -values, which are shown in Section 6 to have more discriminative power.

One of the applications of d-motifs is in finding composite motifs, described in Section 5. (Marsan and Sagot, 2000) present an efficient suffix-tree based algorithm that finds pairs of motifs (with distance constraints) that occur frequently in given promoters. However, raw frequencies should not be the sole criterion for evaluating motifs, since these frequencies have different probability distributions for different motifs. More recently, (Guhathakurta and Stormo, 2001) have developed an algorithm that evaluates composite motifs by an objective function in which both positive and background sequences are considered, and which captures the joint likelihood of occurrence of two simple motifs. However, the motif model used is that of “position weight matrices” or PWMs, whereas our framework is for a different class of models - those where motifs have integral counts in a sequence. Their motif model allows more variability in binding sites, but the motif evaluation assumes that there is exactly one binding site in each input promoter. The d-motifs framework imposes no such restriction. Also, the algorithm of Guhathakurta and Stormo is based on a Gibbs-sampling strategy, and may not find the global optimum in the motif search space.

The most interesting comparison of our work is with that of count based statistical methods such as those in (van Helden et al., 1998), (Tompa, 1999) and (Sinha and Tompa, 2000). Both approaches count motif occurrences in each promoter, and therefore reward a motif that occurs multiple times in the same promoter. This is an important property of the algorithms, since binding sites often occur more than once in a promoter. However, there is an important difference between the approaches. If we want to find motifs whose total count in a set of promoters is surprisingly high, the count based approach is better. On the other hand, if we are looking for overrepresented motifs that are also well distributed over the promoters, d-motifs is the preferable approach. In fact, when restricted to 0 or 1 occurrence per promoter, the count based approach becomes a special case of the d-motif approach. The latter is compared with one of the count based algorithms in terms of performance on synthetic data with planted motifs in Section 6, and is found to do significantly better when the motif occurrences are well distributed.

2 Evaluating motifs

This section describes in detail how a given motif s is evaluated for significance. We shall assume that we are given p “positive” promoters, and n “negative” promoters, each of length L , and the goal is to objectively determine how well the occurrences of s discriminate between the positive and negative promoters. While no assumption is made about what s may look like, we do assume that there is an algorithm to count the occurrences of s in any given promoter, and that the count is an integer. We also assume that the *a priori* probability distribution of the count of s in a randomly generated sequence of length L is known. A parameter to the evaluation is k_{max} , which indicates that k_{max} or more occurrences in a promoter are treated as if there were exactly k_{max} occurrences. While it is desirable to consider motifs with more occurrences in a promoter as more significant, it is reasonable from biological considerations to assume that occurrences beyond a certain large number do not add to the significance of the motif. Moreover, a large motif count in a promoter is sufficiently unlikely that it will not affect our calculations considerably. Additionally, assuming a small value of k_{max} leads to more efficient computations. Our implementation uses $k_{max} = 4$ for motifs of length 6.

We borrow ideas from (Ben-Dor et al., 2000), where feature selection was used to score genes for relevance to tissue classification. Some of the terminology used here is hence borrowed from that work. The first step to evaluating a motif is computing its “TNoM score”, which is defined next. Each positive promoter is plotted as a ‘+’ on the integer line, at the point x if x is the count of s in that promoter. Similarly each negative promoter is plotted as a ‘-’. (See Figure 1.) A “decision stump” is an integer l , and the classification score of a decision stump is the sum of E_n and E_p , where E_n is the number of ‘-’s plotted at $x \geq l$ and E_p is the number of ‘+’s plotted at $x < l$. It is thus the number of errors made by the linear classifier that calls each $x \geq l$ positive. The TNoM (“Threshold Number of Mismatches”) score of s in such a configuration of ‘+’s and ‘-’s is the minimum classification score of any decision stump for that configuration. The significance of s is given by the probability that the TNoM score of s in a configuration derived from random sequences is less than or equal to the TNoM score observed for the configuration derived from the input promoters. This is called the “ p -value score” of s , or simply $pval_s$. We need to use p -values because the TNoM score has a different probability distribution for different motifs.

2.1 Computing the p -value Score of a Motif

A prerequisite to computing $pval_s$ is the knowledge of $p_{s,k} = Pr[\text{count of } s \text{ in a random text of length } L \text{ is } k]$, for $k = 0 \dots k_{max}$. The computation of this distribution depends upon the type of motif s , and is presented in Sections 4 and 5 for two particular types. As mentioned earlier, the TNoM score has been previously used in (Ben-Dor et al., 2000) to score genes for relevance in tissue classification, and the authors present recurrence relations to compute its p -value under slightly different assumptions. We use an alternate set of recurrences to compute $pval_s$, that were preferred over those in (Ben-Dor et al., 2000) for reasons mentioned

shortly.

Let $P(n, p, k_{max}, e)$ denote the probability that the best decision stump for motif s with p positive and n negative promoters, all random sequences, makes e errors. Let the observed TNoM score of s for the configuration derived from the given promoters be TM_s . By definition,

$$pval_s = \sum_{e=0}^{TM_s} P(n, p, k_{max}, e)$$

Also, let $P(n, p, k_{max}, l, e)$ be the probability that of the potentially many decision stumps with the least classification score, l is the rightmost (largest in value) and has a score of e . Then we have

$$P(n, p, k_{max}, e) = \sum_{l=0}^{k_{max}} P(n, p, k_{max}, l, e)$$

Let N_i^+ and N_i^- denote the number of '+'s and '-'s respectively at integer i in a configuration, and let $\delta_i = N_i^+ - N_i^-$. We need to compute the probability that l is the rightmost best decision stump and has score e . It has score e iff i '-'s are placed at $x \geq l$ and $e - i$ '+'s are placed at $x < l$, where $0 \leq i \leq e$. It is the rightmost best decision stump iff two conditions are satisfied:

- Each decision stump to its left has score greater than or equal to its score. This is equivalent to saying that $\delta_{l-1} \leq 0$, $\delta_{l-2} + \delta_{l-1} \leq 0$, ..., $\delta_0 + \delta_1 + \dots + \delta_{l-1} \leq 0$. In what follows, we will need to generalize this condition. Let $S(x, y, low, high, t)$ denote the probability of a (partial) configuration that places x '-'s and y '+'s on the integer line anywhere between integers low and $high$, such that $\sum_{i=min}^{high} \delta_i \leq t$ for all $low \leq min \leq high$.
- Each decision stump to its right has score greater than its score. This is equivalent to saying that $\delta_l > 0$, $\delta_l + \delta_{l+1} > 0$, ..., $\delta_l + \delta_{l+1} + \dots + \delta_{k_{max}} > 0$. Let $T(x, y, low, high, t)$ denote the probability of a (partial) configuration that places x '-'s and y '+'s on the integer line anywhere between integers low and $high$, such that $\sum_{i=low}^{max} \delta_i > t$ for all $low \leq max \leq high$.

Then we can write the recurrence

$$P(n, p, k, l, e) = \sum_{i=0}^e \left[\binom{n}{i} \binom{p}{e-i} S(n-i, e-i, 0, l-1, 0) \right. \\ \left. \times T(i, p-e+i, l, k, 0) \right]$$

Let us now see how we can recursively generate the (partial) configurations whose combined probability is represented by $S(x, y, low, high, t)$. Suppose i '-'s and j '+'s are placed at the integer $high$. The probability of this happening is $p_{s,high}^{i+j}$. There are $x - i$ '-'s and $y - j$ '+'s remaining, that need to be placed in the range $low, \dots, high - 1$. The configurations in the range $low, \dots, high$ are constrained to have $\sum_{i=min}^{high} \delta_i \leq t$

for all $low \leq min \leq high$, and we have $\delta_{high} = (j - i)$. Thus for the partial configurations in the range $low, \dots, high - 1$, we must have $\sum_{i=min}^{high-1} \delta_i \leq (t - (j - i))$ for all $low \leq min \leq high - 1$. The probability of such configurations is represented by $S(x - i, y - j, low, high - 1, t - (j - i))$. Also, the i ‘-’s and j ‘+’s could have been chosen in $\binom{x}{i}$ and $\binom{y}{j}$ ways respectively, and we should allow any possible pairs of values for i and j that can lead to valid configurations. Hence, we have the recurrence

$$S(x, y, low, high, t) = \sum_{i=0}^x \sum_{j=0}^{t+i} \left[\binom{x}{i} \binom{y}{j} p_{s,high}^{i+j} \right. \\ \left. \times S(x - i, y - j, low, high - 1, t - (j - i)) \right]$$

Similarly, we have

$$T(x, y, low, high, t) = \sum_{i=0}^x \sum_{j=t+i+1}^y \left[\binom{x}{i} \binom{y}{j} p_{s,low}^{i+j} \right. \\ \left. \times T(x - i, y - j, low + 1, high, t - (j - i)) \right]$$

The base conditions are:

$$\begin{aligned} S(x, y, low, low, t) &= p_{s,low}^{x+y} \quad \text{if } y - x \leq t \\ &= 0 \quad \text{otherwise} \\ T(x, y, high, high, t) &= p_{s,high}^{x+y} \quad \text{if } y - x > t \\ &= 0 \quad \text{otherwise} \end{aligned}$$

We now analyze the time complexity of a dynamic programming algorithm for computing $P(n, p, k, l, e)$. Computation of $S(x, y, low, high, t)$ requires a table with $\theta(xy(high)t_{range})$ cells, where t_{range} is the size of the range of possible values that t can take in the recursive steps. The value in each cell can clearly be computed in time $\theta(xy)$. Now, notice that the parameter t in the computation of $S(x, y, low, high, t)$ is always non-negative. This is a result of the requirement that $\sum_{i=min}^{high} \delta_i \leq t$ for all $low \leq min \leq high$, and the fact that the initial value of t is 0. Also, for the same reasons, the maximum possible value of t is x . Thus we have $t_{range} \leq x$, and the overall time complexity of computing $S(x, y, low, high, t)$ becomes $O(x^3 y^2 (high))$. Since $S(x, y, low, high, t)$ is called with $x = n - i$, $y = e - i$, $high = l - 1$, and $t = 0$, with $i \leq e$, this time complexity translates to $O(n^3 e^2 l)$, which is $O(n^3 e^2)$ since $l \leq k_{max}$, a constant. A similar argument gives the time complexity of $T(x, y, low, high, t)$ for the specific values of the parameters it is called with as $O(p^3 e^2)$, implying that the overall complexity of computing $P(n, p, k, l, e)$ is $O(e^2(n^3 + p^3))$, as compared to the $\theta(en^2 p^2)$ time that a dynamic programming implementation of the recurrences adapted from (Ben-Dor et al., 2000) would take. Since it is often the case that the algorithm needs the probabilities only for $e = o(max(n, p))$ (See Section 3), and since in our algorithm p is equal to n , our set of recurrences is more efficient asymptotically. Another advantage of these recurrences is that they may be extended to

work when the configurations are on the 2-dimensional integer grid, rather than the integer line. This case arises when considering some models of composite motifs. The recurrences in (Ben-Dor et al., 2000) have no natural extension to the 2-dimensional case. It is an open problem to compute the extensions of our recurrences efficiently by using suitable approximations.

3 Algorithm

The algorithm described here is given a set P of p sequences of length L each, and the goal is to find the best motifs that discriminate the input sequences from random sequences. We assume a well-defined random process R that generates sequences of A,C,G,T in a way that mimics typical background sequences. Other parameters include an integer, *numbags*, which is explained later, and an integer k_{max} described in Section 2. The algorithm is described without assuming any particular motif model, but it is required that some finite set S enumerates all motifs satisfying the model. The algorithm outputs the motifs in S sorted by their significance.

Algorithm DMotifs

1. For $i = 1$ to *numbags* do
2. Create a set N of p random strings (length L each), using the random process R .
3. For each motif s in S do
4. Compute the TNoM score of s using P and N as the positive and negative sets of promoters.
5. Compute $pval_s$ using the algorithm described in Section 2.1.
6. End
7. Sort the motifs in S in non-decreasing order of $pval_s$ and call this sorted list Λ_i .
8. End
9. For all motifs s in S do
10. Compute the combined rank of s as the sum of its ranks from each Λ_i .
11. End
12. Sort all motifs in S by their combined rank in non-decreasing order and report this sorted list.

End Algorithm DMotifs

Steps 2 to 7 implement one iteration of d-motif finding, using the p input promoters as positives and an equal number of randomly generated sequences as negatives. The loop 1...8 repeats this iteration *numbags* times, each time using a different set of negatives, and thus implements a simple form of “bagging”, a technique popular in machine learning. This approach is chosen instead of having one iteration with a larger set of negatives because the time complexity of computing $pval_s$ is cubic in the number of negatives. An important feature of the algorithm is that it makes no commitment to the exact search space S used, and can therefore be used with any motif model and any heuristic search that traverses only part of the search space. For efficiency, the DMOTIFS implementation computes the p -value score (Step 5) only if the TNoM score is below a certain threshold; otherwise the motif is rejected as being insignificant, and is not included in the output list Λ_i for that iteration.

4 Simple motifs

This section describes a simple application of the DMOTIFS algorithm. The motif model used is that of consensus sequences with IUPAC degenerate symbols. In particular, each motif in the space is a string over the alphabet $\{A, C, G, T, R, Y, S, W, K, M, N\}$, with each of R, Y, S, W, K, M representing a disjunction of two bases, and N representing a spacer (any base). The spacers may occur only in the middle of a motif, and the number of degenerate symbols is typically restricted to some small constant. This model, henceforth called the *consensus model*, has been successful in describing transcription factor binding sites in *S. cerevisiae* (Zhu and Zhang, 1999) and has been used in other algorithmic approaches, especially enumerative ones such as (Tompa, 1999) and (Sinha and Tompa, 2000), to detect binding sites in that organism. The typical length of the string, excluding the spacers, is 6-8.

To use this model in the DMOTIFS framework, we need to compute $p_{s,k}$, the probability distribution of the count k of a motif s in a randomly generated sequence of length L . We adopt a simulation-based approach to estimate this probability distribution, since existing methods are either too inefficient to be run on all motifs in the model or are incompatible with the model. (See Section 7.) Each simulation generates a random text of length L and counts the occurrences of s as specified by the model. We shall now compute how many simulations need to be done to get a desired degree of accuracy in estimating the probability distribution.

Suppose that N such simulations are done, and let X be the number of simulated sequences that yielded a count of k . Then we estimate $p_{s,k}$ as being X/N . To bound the absolute error in computing $p_{s,k}$ by c_1 , we need $|p_{s,k} - X/N| \leq c_1$, i.e., $|X - Np_{s,k}| \leq Nc_1$, and we shall say that $p_{s,k}$ is estimated incorrectly if this condition is violated. We know that X is a binomial variable with parameters N and $p_{s,k}$. Hence, using Chernoff's bound (McDiarmid, 1998), we have

$$Pr[|X - Np_{s,k}| > Nc_1] \leq 2e^{-2Nc_1^2}$$

Thus, $Pr[p_{s,k}$ is estimated incorrectly] $\leq 2e^{-2Nc_1^2}$. If we wish to compute $p_{s,k}$ for all s in some set S , and for all $k \in \{0 \dots k_{max} - 1\}$, we can claim that $Pr[p_{s,k}$ is incorrect for some $s] \leq 2|S|k_{max}e^{-2Nc_1^2}$. Also suppose we want to bound the probability of making *any* incorrect estimate at all by c_2 . This is achieved if we have $2|S|k_{max}e^{-2Nc_1^2} < c_2$, which is the same as

$$N > \frac{\ln(2|S|k_{max}/c_2)}{2c_1^2}$$

This relation is used in deriving the number of simulations that are done. For instance, if $c_1 = 10^{-3}$, $c_2 = 10^{-2}$, $k_{max} = 4$ and S contains all motifs of length 6 in the consensus model, with at most 1 degenerate symbol and no spacers ($|S| = \binom{6}{0}6^04^6 + \binom{6}{1}6^14^5$), we have $N > 8.66 \times 10^6$. A crucial feature of the relation is that N grows only logarithmically with the size of S , which is roughly linear in the length of the motifs. Our implementation uses $c_1 = 0.001$ and $c_2 = 0.01$. Since we shall be interested in computing $p_{s,k}$ only for small values of k (typically less than 4), it is reasonable to use such relatively large values of c_1 .

The count of a motif in a promoter includes overlapping occurrences, and occurrences in either strand.

This can be easily computed in $O(L)$ time. The random text is generated according to a 3^{rd} order Markov model that is trained on all promoters of the organism under study (yeast, in our case). Now, since we have a well defined motif search space, an algorithm to count motif occurrences, and accurate estimates of the probability distributions of motif counts, we can invoke the DMOTIFS framework of Section 3 for finding simple motifs.

4.1 Validation: Results on known regulons

We ran DMOTIFS on known regulons, which are sets of genes known to be co-regulated, and where the binding site has been biologically characterized. SCPD catalogues such regulons in yeast, along with the binding site consensus sequence for each. DMOTIFS was run on *all* regulons in SCPD that have at least 3 genes, and have a consensus sequence for the binding site. The motif length (excluding spacers) parameter was varied between 5 and 8. The maximum number of spacers allowed was 6 (except for the GAL4 regulon, when this was 11) for motif length 6, and 0 for other motif length values. The decision to allow a maximum of 6 spacers for a motif length of 6 was purely for the sake of efficiency of the experiments. In all runs, a maximum of 1 degenerate symbol was allowed. Table 1 describes the results for the best choice of the motif length parameter. We see that in 18 out of the 23 regulons, the known consensus closely matches one of the top 10 motifs reported by the algorithm, and in 15 of these 18, the match is in the top 2.

The regulon ROX1 is an example of a count based approach leading to a false positive. As we see in Table 1, DMOTIFS finds the motif CCTATTG, which matches the binding site closely, at rank 7. However, if we run the algorithm YMF of (Sinha and Tompa, 2000), which measures overrepresentation based on total counts, the top motif is TTYTTTT, and all of the best 20 motifs are variations of this motif. TTYTTTT has 58 occurrences in the three genes, whereas the expectation is 8.9 per gene, so the total count is significantly high. A closer examination reveals that the 58 occurrences are distributed as 10,11 and 37 in the 3 genes. Thus in two of the genes, the count is close to expectation, and the total count is boosted by the third gene. DMOTIFS is able to recognize this phenomenon and does not report the motif, while a count based approach like YMF incorrectly reports it as significant.

4.2 Details of Simulations

In this section, we describe the implementation of the program that simulates sequences to estimate the probability distributions $p_{s,k}$ of motif counts. An important feature of the implementation is its cache-consciousness. We focused on this aspect because this is the most time-consuming component of the DMOTIFS framework, and it was particularly important to have an efficient implementation. Let the cache line size be B . This is the number of bytes that are fetched from memory to the cache when any memory request is made. The key idea is to increase locality of reference of the data structures to reduce memory accesses. There are three main data structures, each being an array. See Figure 2 for an overview of their organization. The first data structure, a two-dimensional array called TALLIES, has one row for each *basic* motif (a motif

without any degenerate symbols). For instance, there are $4^8 = 64K$ basic motifs of length 8. Each row is in turn an array of B bytes, and each cell in this array counts the occurrences of a basic motif in one simulated sequence. The second data structure, an array called INDICES, has one entry for each motif (possibly with degenerate symbols). The entry for motif m is a list of pointers to the rows in TALLIES that correspond to motifs that are instantiations of m . The third data structure, a two-dimensional array called DISTRIBUTIONS, has one row for each motif. The row for any motif m is an array of size $k_{max} + 1$, and the i^{th} cell of this array counts the number of simulated sequences encountered so far that had $i - 1$ occurrences of motif m . (The $k_{max} + 1^{th}$ entry counts how many sequences had k_{max} or more occurrences of m .) After all N simulations have been done, the row for motif m in DISTRIBUTIONS can be used to compute the estimated probability distribution of the count of m .

The N simulations are done in sets of B . In each iteration of a loop, B sequences (of the input length L) are generated according to the background model. In a single pass of each sequence, the count of each basic motif is updated in the appropriate cell in TALLIES. (This can be done in time linear in the length of the sequences by exploiting the fact that each basic motif of length l can be mapped to a unique $2l$ -bit number, which is used as the index of that motif in the array TALLIES. This $2l$ -bit number can be computed in constant time from the number for the motif that starts one position to the left of this motif, and the character at its right-most position.) Then, for each motif m , the array INDICES is used to fetch the list of basic motifs that correspond to m . The rows in TALLIES corresponding to this list of motifs (each entry being an array of B bytes) are then added as vectors to produce a single vector of B bytes that represents the frequencies of motif m in the B simulated sequences. The row for m in DISTRIBUTIONS is then updated with this information. After $\lceil N/B \rceil$ iterations of this loop, at least N simulations have been done, and each row of DISTRIBUTIONS can be normalized to give the probability distribution for the corresponding motif.

The most memory-intensive step of the algorithm above is where the rows in TALLIES corresponding to a motif m are added, for each motif m . Fetching each row corresponds to B memory accesses, but since the cache line size is B , the entire row is fetched into the cache when its first entry is accessed. This could save up to $B - 1$ cache misses in a pathological scenario. We compared the performance of the algorithm to another implementation in which each row of TALLIES contains a single byte (instead of B bytes). For motifs of length 6, with at most 1 degenerate symbol and no spacer, we observed that there was a 4-fold increase in running time in the latter implementation. The performance hit increases as the number of basic motifs increases.

5 Composite motifs

For some time now, there have been attempts to model and detect *combinations* of motifs that occur in promoter regions; for instance, (Grundy et al., 1997), (GuhaThakurta and Stormo, 2001), (Hu et al., 2000) and (Marsan and Sagot, 2000). Such a motif is the footprint of an interaction of multiple transcription factors and other regulatory proteins, found on the promoter, and is often called a “higher order” motif or a “composite” motif. For instance, in the yeast *S. cerevisiae*, the transcription factors MCM1 and STE12 are known to jointly regulate the basal expression of genes FAR1 and STE2, with the corresponding promoters having clustered binding sites for both factors (Wagner, 1999). Similarly, the promoters of the eight core histone genes have a conserved pattern $m_1 - d_1 - m_2 - d_2 - m_2$, where m_1 and m_2 are well characterized simple motifs, d_1 is a tight interval of spacers, and d_2 is a less tight interval (Pavlidis et al., 2000). Composite motifs in other eukaryotes are known to be more complex (Arnone and Davidson, 1997).

5.1 Application of DMOTIFS to Detect Composite Motifs

Here we show the application of the DMOTIFS algorithm to the detection of composite motifs. A composite motif s is modeled as a triplet (s_1, s_2, d) , where s_1 and s_2 are simple motifs following the consensus model, and s is said to occur when s_1 and s_2 occur, on either strand and in either order, with their starting positions within d bases of each other. This is a simple “paired motif” model that has support from the biology literature, for instance (Ohmori et al., 1997). Additional constraints may be imposed to obtain a different model to which d-motif finding can then be applied, thereby exploiting the generality of the framework. To use DMOTIFS, we need to specify the search space of motifs, a method to count motif occurrences, and the *a priori* probability distribution of motif counts, all of which will be described now.

Let S be the set of simple motifs used in Section 4. For the motif space, we use the set $S' = \{(s_1, s_2, d) | s_1 \in S, s_2 \in S, d = \text{a fixed constant}\}$. Given a triplet $s = (s_1, s_2, d)$, a sequence is said to have k occurrences of s if there are k “pairs” in the sequence, each pair comprising an occurrence of s_1 and an occurrence of s_2 separated by d or less positions, no two of the k pairs sharing an occurrence of either s_1 or s_2 , and k being the maximal such number. For efficiency, the implementation searches a subset of the motif space S' . It first computes the set of the best t simple motifs (according to their p -value scores), where t is a parameter (set to 2000), and then searches the space of all pairs of motifs from this set, with a fixed value of d . The search is also restricted to pairs that do not have a high overlap (more than half their length) in their sequences.

We compute $p_{s,k} = Pr[\text{count of } s \text{ in a random text of length } L \text{ is } k]$ by making some simplifying assumptions. We assume that the counts of s_1 and s_2 are independent. Then

$$p_{s,k} = \sum_{k_1=k}^{k_{max}} \sum_{k_2=k}^{k_{max}} p_{s_1,k_1} p_{s_2,k_2} p_{k_1,k_2,k}$$

where $p_{k_1,k_2,k}$ is the probability that there are k occurrences of the paired motif s given that there are k_1 occurrences of s_1 and k_2 occurrences of s_2 . If we further assume that all choices of positions of the k_1

occurrences of s_1 and k_2 occurrences of s_2 are equally likely, then computing $p_{k_1, k_2, k}$ amounts to solving a particular balls-and-urns problem, and can be done irrespective of the patterns s_1 and s_2 .

The BALLS_AND_URNS problem is: *Given that k_1 black balls and k_2 white balls are placed in a sequence of L urns ordered from left to right, at most one ball in an urn, how many distinct configurations or placements have at least k “pairs”, each of which includes a black and a white ball, such that the number of urns between the two balls of each pair is less than or equal to d ? The k pairs are non-overlapping - no two pairs share a ball and no ball in a pair lies between the balls of another pair.*

Consecutive urns in the above problem correspond to consecutive positions in the sequence, and placements of black and white balls indicate starting positions of s_1 and s_2 respectively. If the answer to the above problem is x , then of all the distinct ways of placing k_1 occurrences of s_1 and k_2 occurrences of s_2 in a sequence of length L , exactly x configurations have k or more occurrences of the paired motif $s = (s_1, s_2, d)$. Thus the number x can be used to compute the probability $p_{k_1, k_2, k}$.

It can be shown (proof not shown here) that the following process P' can be used to systematically generate all distinct configurations described in the above problem. On inputs $low, high, k_1, k_2, k$, it places k_1 black balls and k_2 white balls between (and including) positions low and $high$ such that at least k “pairs” with the specified restrictions are formed.

Process P' :

Inputs: $low, high, k_1, k_2, k$

1. If $k = 0$, place the $k_1 + k_2$ balls between low and $high$ in all possible ways. Return.
2. Choose a pair of positions π_1, π_2 with $low \leq \pi_1 < \pi_2 \leq high$ and $\pi_2 - \pi_1 \leq d + 1$, in all possible ways. Place a black ball at π_1 and a white ball at π_2 , or vice versa.
3. Choose two numbers $x \leq k_1 - 1$ and $y \leq k_2 - 1$ in all possible ways. Place, if possible, x black balls and y white balls in the range $low \dots (\pi_1 - 1)$ in all possible ways, making sure that no pair (of black and white balls, with less than $d + 1$ urns between them) whose right ball is at or to the left of π_1 , is formed by this placement.
4. Repeat process P' on inputs $(\pi_2 + 1, high, k_1 - 1 - x, k_2 - 1 - y, k - 1)$.

End Process P'

By making different choices in each step, this process, when called with inputs $0, L - 1, k_1, k_2, k$, can generate exactly the set of configurations mentioned in the BALLS_AND_URNS problem. The process is recursive, and the number of configurations it generates can be counted by a dynamic programming algorithm, as follows.

Let $S(l, k_1, k_2, k)$ be the number of ways to place k_1 black balls and k_2 white balls in the range of positions $0 \dots l$, such that at least k pairs with the specified restrictions are formed. Let $X(l, k_1, k_2)$ be the number

of ways to place k_1 balls of color c_1 and k_2 balls of color c_2 in the range $0 \dots l$, such that no valid pair of c_1 -colored and c_2 -colored balls is formed in the range $0 \dots (l+1)$, given that a ball of color c_1 is placed at position $l+1$. The placement procedure corresponding to $S(l, k_1, k_2, k)$ first chooses a position π_1 and a separation $\delta \leq d$ and places a pair at positions π_1 and $\pi_1 + \delta + 1$. This is the leftmost pair that will be formed satisfying the specified restrictions. It then chooses two numbers x and y , and places x black balls and y white balls to the left of π_1 without creating any new pairs. If a black ball was placed at π_1 , this can be done in $X(\pi_1 - 1, x, y)$ different ways, and in $X(\pi_1 - 1, y, x)$ ways if a white ball was placed at π_1 . Once these placements have been done, $k_1 - x - 1$ black balls and $k_2 - y - 1$ white balls are placed in the range $\pi_1 + \delta + 2, \dots, l$ to form at least $k - 1$ pairs, and this can be done in $S(l - \pi_1 - \delta - 2, k_1 - x - 1, k_2 - y - 1, k - 1)$ ways. Hence the recurrence for S is:

$$S(l, k_1, k_2, k) = \sum_{\pi_1=0}^{l-2k+1} \sum_{\delta=0}^d \sum_{x=0}^{k_1-1} \sum_{y=0}^{k_2-1} ([X(\pi_1 - 1, x, y) + X(\pi_1 - 1, y, x)]) \times S(l - \pi_1 - \delta - 2, k_1 - x - 1, k_2 - y - 1, k - 1)$$

The important base condition is:

$$S(l, k_1, k_2, k) = \binom{l+1}{k_1+k_2} \binom{k_1+k_2}{k_1} \quad \text{if } k = 0$$

Other base conditions are trivial, and are omitted here. We now present the recurrence for X . The placement procedure is recursive, placing balls one at a time, from right to left. The first ball placed to the left of $l+1$ could be c_1 colored or c_2 colored. If it is c_1 colored, then it could be placed at any position to the left of $l+1$ without violating the required conditions, since a c_1 colored ball is at $l+1$. However, if it is a c_2 colored ball, then it should be at least $d+1$ urns away from the c_1 colored ball at $l+1$, hence it has to be placed at or to the left of $l-d-1$. Once the first ball to the left of $l+1$ is placed at some position u , the same procedure can be applied to place the remaining balls in the region to the left of u . Hence, the recurrence for X is:

$$X(l, k_1, k_2) = \sum_{u=k_1+k_2-1}^l X(u-1, k_1-1, k_2) + \sum_{u=k_1+k_2-1}^{l-d-1} X(u-1, k_2-1, k_1)$$

The important base conditions are:

$$\begin{aligned} X(l, k_1, k_2) &= \binom{l-d}{k_2} \quad \text{if } k_1 = 0 \wedge (0 \leq k_2 \leq l-d) \\ &= \binom{l+1}{k_1} \quad \text{if } k_2 = 0 \wedge (0 \leq k_1 \leq l+1) \end{aligned}$$

The probability that there are at least k pairs formed is then given by

$$\frac{S(L-1, k_1, k_2, k)}{\binom{L}{k_1+k_2} \binom{k_1+k_2}{k_1}}$$

The probability $p_{k_1, k_2, k}$ of forming exactly k pairs is then simply the difference of two such expressions. We now analyze the time complexity of the dynamic programming algorithms that implement the above recurrences. The computation of $S(L - 1, k_1, k_2, k)$ uses a table with Lk_1k_2k entries, and each cell of the table can be computed from the recurrence in time $O(Lk_1k_2d)$. Hence the time complexity of computing $S(L - 1, k_1, k_2, k)$ is $O(L^2k_1^2k_2^2kd)$. By a similar analysis, the time complexity for computing $X(l, k_1, k_2)$ is $O(l^2k_1k_2)$, which for $l \leq L$ is $O(L^2k_1k_2)$. Hence the overall time complexity of the computation is $O(L^2k_1^2k_2^2kd)$. The program has to be run only once for fixed values of L and d , and all practical values of k_1, k_2, k .

5.2 Application to biological data

The composite motif application of DMOTIFS was validated on a biological data set. Transcription factors MCM1 and STE12 are known to jointly regulate the expression of gene FAR1. SCPD catalogues three other genes MFA1, MFA2 and STE2 in which the binding sites of both MCM1 and STE12 are known to occur. We took these four genes (FAR1, MFA1, MFA2 and STE2) and ran DMOTIFS on them, to look for composite motifs. The search space included all pairs of 6-mers, with no spacers or degenerate symbols, separated by up to 40 bases. The parameters used in the algorithm were $k_{max} = 4$ and $numbags = 10$. The 2^{nd} most significant motif reported was the pair (TGAAAC, GGAAAT). We noted that TGAAAC is a close match to the STE12 consensus ATGAAA, and GGAAAT occurs overlapping with MCM1 and MATAALPHA2 binding sites in these four genes. While TGAAAC was ranked 2^{nd} among the simple motifs in this regulon, GGAAAT was ranked 56^{th} and would not have been deemed significant had it not been for the composite motif it is a part of. The top ranking composite motif was (GGAAAT, TACATG). We note that GGAAAT occurs overlapping with MCM1 binding sites and TACATG is a part of the MATAALPHA2 binding sites known to occur in the promoters under study. We also noted that MCM1 is known to interact with MATAALPHA2 for purposes of transcriptional regulation (Mewes et al., 1999).

6 Experiments on synthetic data

We now describe experiments where the DMOTIFS algorithm was run on synthetic sequences with planted motifs. Ten sequences, of length 800 each, were generated according to a random process that mimics upstream sequences in yeast. A simple motif s of length 6 was chosen at random. We computed its expected count E_s in 10 random sequences, and planted it at $r \times E_s$ random places in the 10 synthetic sequences. (r is the “overrepresentation” factor, a parameter to the experiments. The results discussed below are for $r = 2.5$.) The planting was done in a way that distributed the instances as evenly as possible across the sequences. This set of sequences was then fed as input to the motif finding algorithm under study, and the algorithm was made to report motifs in decreasing order of significance. One algorithm is said to *win* against another if it reports s at a higher (better) rank than the other. The difference in the ranks of s in the two reported lists is called the *margin*.

In the first experiment, algorithm DMOTIFS was compared with the algorithm YMF (Sinha and Tompa, 2000), a count based algorithm. The above experiment was run 500 times, each time with a different randomly chosen motif being planted in newly generated random sequences. Both algorithms were run with the same parameters – 0 spacers and 0 degenerate symbols. DMOTIFS won against YMF on 285 of the 500 trials, YMF winning 49 times, and the remaining 166 were ties. We repeated the same experiment, but now with motif occurrences being planted independently at random. YMF outperformed DMOTIFS comprehensively, winning 343 and losing 69. These experiments demonstrate the strength of the DMOTIFS approach when motif instances are well distributed over the promoters, as one would expect in certain applications with real data.

In another experiment, two variants of DMOTIFS were compared - one that uses the p -value score and one that uses the TNoM score for motif significance. Out of 500 trials (as above), the p -value variant won in 136 and the TNoM score variant won in 51, the rest being ties. The average margin for the former’s wins was 14, while that for the latter was 2. The scale is tipped further in favor of the p -value variant when the overrepresentation factor r is reduced. These results provide clear experimental evidence that the p -value score has more discriminative power. In a similar experiment, we studied the effect of bagging on the performance. The two variants of DMOTIFS that were compared used $numbags = 10$ and $numbags = 1$ respectively. The former won 260 times out of 500, and lost 48 times, thereby making a strong case for using the bagging heuristic. In yet another heuristic-evaluation experiment, we implemented DMOTIFS to use as negative promoters a randomly chosen subset of all yeast promoters, instead of the the randomly generated sequences that the standard implementation uses. Both variants performed equally well, and out of 500 trials, 260 were ties, with the two variants winning 122 and 118 times respectively.

We also tested the composite motif finder on synthetic data. 10 sequences were created as above. Two patterns of length 6 each were chosen at random, each with up to 6 spacers and 1 degenerate symbol. This pair was planted at random positions in 7 of the 10 sequences. When planting the pair, the distance between the two patterns was chosen at random from a Poisson distribution with mean 15. DMOTIFS was run on these

10 sequences, and configured to look for pairs of simple motifs of length 6 (each with up to 6 spacers and 1 degenerate symbol), separated by at most 40 bases. The entire experiment was repeated 10 times, each time with a different pair of motifs. In each case, the planted pair or a close variant was recovered as the top ranking motif. However, this in itself does not measure the performance of the algorithm in any quantitative sense, since planting the motif in fewer or more sequences will change the performance accordingly. On the other hand, the scores of the motifs do tell us something interesting about the evaluation criteria. In one of the 10 experiments, we planted the pair (GTGTTC,AGTGCT) and examined, for one of the bagging iterations, the scores of all motifs in the search space. The planted motif was recovered as the top ranking motif based on its p -value score. It had a TNoM score of 3. At the same time, over 50% of the top 1000 composite motifs reported were found to have a TNoM score of 3 or less. Many of these were very different from the planted motif, and had a better TNoM score merely by chance. This result strongly argues for preferring p -value scores over TNoM scores. When exploring a large motif space, it is very likely that some spurious motifs will score equally well with TNoM. The same argument holds against using raw counts of motifs as the evaluation criterion.

7 Discussion

We first discuss why simulations were used in Section 4 to estimate the probability distribution $p_{s,k}$ of the counts of simple motifs, instead of a potentially faster analytical method. The motif count includes overlapping occurrences of the motif, and occurrences in either strand. It is thus the sum of the indicator variables for the presence or absence of the motif at each position of the promoter sequence, on either strand. These indicator variables have intricate dependencies between them depending upon the self-overlap structure of the motif, and hence lead to non-trivial analysis. One approach to computing such probabilities is by demonstrating that the count approximately follows a compound Poisson distribution, and computing the parameters of this distribution. This method is described in (Waterman, 1995), for single stranded occurrence. However, it is not obvious if and how the procedure can be efficiently extended to handle degenerate symbols and double stranded occurrence, both of which complicate the dependencies between occurrences. (Nicodème et al., 1999) has shown how to compute probability distributions for motif counts under a large class of models, which includes the consensus model we use, but their method is not efficient enough to be run for all motifs in our search space. Thus it is not clear how the distributions can be computed analytically for the motif model used here, and simulations are a viable option, being a one-time operation.

7.1 Choice of Scoring Function

The main contribution of the DMOTIFS framework presented here is the use of a new scoring function, the p -value of the TNoM score, which measures overrepresentation without ignoring the distribution of motif instances in the input promoters. Given two sets of sample points (the motif counts in the positive and negative promoters respectively), the p -value of the TNoM score measures, in a sense, how likely it is that the two sample sets came from the same distribution. Lower p -values indicate lower probability of this happening. There are several statistical measures for determining if the sets could have been sampled from the same distribution. We shall now discuss some of these statistics and their pros and cons for our application.

1. **t -score:** This is a popular *parametric* statistic that is designed to test if two sample sets come from the same distribution, assuming that the underlying distribution is normal. It measures the difference between the sample means, normalized by their variances. A sufficiently high value of this statistic indicates that the sample means are too different for the samples to have come from the same distribution. However, this statistic may not be ideal for assessing the difference of mean motif counts in positive and negative promoters, for the following reasons:
 - motif counts cannot be assumed to be normally distributed for typical values of the promoter length (~ 1000).
 - the t -score is not robust to outliers, and one or a few very large sample points may completely change the measure. Such outliers are not unusual with motif counts.

- the t -score compares total counts and is not very good at differentiating well distributed motif instances from skewed distributions, although the variance term in the denominator (see Equation 1 below) does cause well distributed motifs to score better than motifs with skewed distribution of occurrences.

We verified our intuition about the t -score by re-implementing DMOTIFS so as to use the t -score instead of the TNoM p -value score, and comparing its performance with the original version. The formula for the t -score is given below. If the positive samples are x_1, x_2, \dots, x_p and the negative samples are y_1, y_2, \dots, y_n and the two sample means are μ_1 and μ_2 respectively, then

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{ss}{p} + \frac{ss}{n}}} \quad (1)$$

where

$$ss = \frac{\sum_{i=1}^p (x_i - \mu_1)^2 + \sum_{i=1}^n (y_i - \mu_2)^2}{n + p - 2} \quad (2)$$

Notice that the numerator is $\mu_1 - \mu_2$, instead of $|\mu_1 - \mu_2|$ that occurs in the numerator of the standard versions of the t -score. This is intended to capture how much *higher* the motif counts in the positive promoters are than in the negative promoters, which is what our application demands. The comparison between the two variants of DMOTIFS (one using the TNoM p -value and the other using the t -score) was done in the manner described in Section 6. 5 – 20 sequences of length 800 each were generated according to the background Markov model, and a random motif was planted in these several times more than expectation, the motif instances being distributed uniformly across the ten promoters. Both variants of DMOTIFS were run on this data set, and the ranks at which these two reported the planted motif were compared. The algorithm variant that reports the planted motif at a lower (better) rank is said to win. This experiment was repeated with 500 different motifs, and we counted how many times each variant won, lost or tied. Table 2 summarizes the observations. Notice that for larger number of input promoters, the TNoM p -value variant does better than the t -score variant. The performance is comparable for small number of sequences.

Note that we need not use the p -value of the t -score since the p -value is monotonic with the t -score, and does not differ from one motif to another.

2. **Non Parametric Test Statistics:** These statistics compare two samples to determine if they come from the same distribution, without making assumptions about the underlying distribution. Apart from the fact that non-parametric scores do not assume normality, another attractive feature of these is that they are more robust to outliers than say the t -score. One popular non-parametric score for comparing two sample sets is the Wilcoxon Rank Sum Test score, which we shall henceforth call the *RST*-score. To compute this score, we first sort all samples (positive as well as negative) in ascending order of value, and then assign a rank to each sample, the lowest one getting rank 1 and each successive

sample getting a rank 1 more than the previous. Tied sample values share the same rank, which is the average of their positions in the sorted list. The ranks of the positive samples are averaged and so are the ranks of the negative samples, and the difference in these two rank-averages is the desired *RST*-score. The intuition behind this statistic is that if the positive and negative samples came from the same distribution, their rank-averages will not differ significantly. The *RST*-score is a popular non-parametric alternative to the *t*-score. Moreover, it is believed that the *RST*-score is a more powerful statistic than the TNoM score in the absence of ties, which is the case, for instance, when the samples are real valued rather than integers. However, in our application, the underlying distribution (motif counts in random sequences) is discrete, and ties are common even with modest numbers of samples. Moreover, the probabilities of ties depend upon the specific motif. Hence a non-parametric score that works well in the absence of ties may not work well in this scenario, especially when compared to a score that uses knowledge of the underlying distribution. Thus the *RST*-score, as well as the raw TNoM score, is at a disadvantage in comparison to the TNoM *p*-value score, since the latter uses motif-specific distributions. The *p*-value of the *RST*-score, if computed based on knowledge of the underlying distribution, may perform comparably, or even better, but we do not have an efficient method to compute these *p*-values. Hence the TNoM *p*-value score emerges as the preferable scoring function.

We confirmed this intuition by comparing the performance of the *RST*-score with that of the TNoM *p*-value score, in a manner identical to the comparison with the *t*-score that is described above. Two variants of DMOTIFS, using the two scores respectively, were compared with respect to their performance on simulated data with planted motifs. The results are summarized in Table 3. The TNoM *p*-value score consistently outperforms the *RST*-score, and the gap increases with larger numbers of input promoters.

In summary, the merits of the TNoM *p*-value score, due to which it was chosen as the scoring function in DMOTIFS, are:

1. The probability distribution of motif counts, which is different from one motif to another, is used in computing the score.
2. No assumption, such as normality, is made about the distribution.
3. The score is robust to outliers.
4. The score is based on the relative values of all the positive and negative samples, and not just on some combined property of the samples, such as mean or rank-sum.

8 Future work and conclusion

As mentioned earlier, algorithm DMOTIFS does not assume any specific motif model, or search technique. It only specifies how motifs are evaluated. Depending upon the specific motif model used, the search technique may also be modified. Step 3 in the algorithm of Section 3 traverses the space S in an arbitrary manner. An algorithm such as that of (Marsan and Sagot, 2000), which can output composite motifs in the order of their count in the input regions, may be used instead, in adjunction with a heuristic that rejects motifs with low counts without computing their $pval_s$. This is foremost on our agenda for future work. Another issue worth investigating is how to decide upon the significance of p -value scores. Currently, we are only able to *rank* motifs by their scores. One of the tasks cut out for the immediate future is the application of the framework to other motif models, such as the mismatch model. A broad and important direction of research is to extend the DMOTIFS technique to work with non-integral counts. This would allow more flexibility in the notion of motif occurrence. A shortcoming of the DMOTIFS algorithm is that it does not scale well with the number of promoters – its time complexity is cubic in this number. Improvements in this direction are needed to make the algorithm feasible for large sets of input sequences.

This work presents a new characterization of the overrepresentation of a motif, and develops a motif discovery algorithm under this characterization. The general algorithm is adapted for two specific motif models, and shown to work well on real as well as synthetic data.

9 Acknowledgments

This material is based upon work supported in part by the National Science Foundation under grant DBI-9974498 and in part by Microsoft under a Microsoft Research Fellowship. The author wishes to thank his advisor, Martin Tompa, for his active support through regular discussions. The contribution of Mathieu Blanchette, who had insightful comments throughout the project, cannot be overstated. A very useful discussion with Uri Keich (University of California, San Diego) on the merits and weaknesses of the various scoring functions is gratefully acknowledged. The author also thanks the referees for their useful suggestions, and Rimli Sengupta for the discussions on the subject.

References

- Arnone, M. I. and Davidson, E. H. 1997. The hardwiring of development: organization and function of genomic regulatory systems. *Development* 124, 1851–1864.
- Bailey, T. L. and Elkan, C. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning* 21, 51–80.
- Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M. and Yakhini, Z. 2000. Tissue classification with gene expression profiles. *Journal of Computational Biology* 7, 559–584.
- Grundy, W. N., Bailey, T. L., Elkan, C. P. and Baker, M. E. 1997. Meta-meme: Motif-based hidden markov models of protein families. *Computer Applications in the Biosciences* 13, 397–406.
- GuhaThakurta, D. and Stormo, G. D. 2001. Identifying target sites for cooperatively binding factors. In *RECOMB01: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology*. Montreal, Canada.
- Hertz, G. Z. and Stormo, G. D. 1995. Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps. In Lim, H. A. and Cantor, C. R., eds., *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, 201–216. World Scientific Publishing Co., Ltd., Singapore.
- Hu, Y.-J., Sandmeyer, S., McLaughlin, C. and Kibler, D. 2000. Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics* 16, 222–232.
- Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F. and Wootton, J. C. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262, 208–214.
- Marsan, L. and Sagot, M.-F. 2000. Extracting structured motifs using a suffix tree - algorithms and application to promoter consensus identification. In *RECOMB00: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, 210–219. Tokyo, Japan.
- McDiarmid, C. 1998. Concentration. In Habib, M., McDiarmid, C., Ramirez-Alfonsin, J. and Reed, B., eds., *Probabilistic Methods for Algorithmic Discrete Mathematics*, 195–248. Springer.
- Mewes, H., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S. and Frishman, D. 1999. Mips: a database for protein sequences and complete genomes. *Nucleic Acids Research* 27, 44–48.
- Nicodème, P., Salvy, B. and Flajolet, P. 1999. Motif statistics. Technical Report RR-3606, INRIA Rocquencourt.
- Ohmori, Y., Schreiber, R. D. and Hamilton, T. A. 1997. Synergy between interferon-gamma and tumor necrosis factor alpha in transcriptional activation is mediated by cooperation between signal transducer

- and activator of transcription 1 and nuclear factor kappa b. *The Journal of Biological Chemistry* 14899–14907.
- Pavlidis, P., Furey, T., Liberto, M., Haussler, D. and Grundy, W. 2000. Promoter region-based classification of genes. *Pacific Symposium on Biocomputing* .
- Roth, F. P., Hughes, J. D., Estep, P. W. and Church, G. M. 1998. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology* 16, 939–945.
- Sinha, S. and Tompa, M. 2000. A statistical method for finding transcription factor binding sites. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press.
- 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*, 262–271. AAAI Press, Heidelberg, Germany.
- van Helden, J., André, B. and Collado-Vides, J. 1998. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology* 281, 827–842.
- Wagner, A. 1999. Genes regulated cooperatively by one or more transcription factors and their identification in whole eukaryotic genomes. *Bioinformatics* 15, 776–784.
- Waterman, M. S. 1995. *Introduction to Computational Biology*. Chapman & Hall.
- Zhu, J. and Zhang, M. Q. 1999. SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics* 15, 563–577. <http://cgsigma.cshl.org/jian/>.

REGULON	BINDING SITE	MOTIF FOUND	RANK	PARAMETERS
ABF1	TCRNNNNNNACG	TCANNNNNNAMG	2	6,6
CPF1	TCACGTG	CACGTG	1	6,6
CSRE	YCGGAYRRRAWGG	CGGATGRA	8	8,0
SCB	CNCGAAA	TCGCGAA	2	7,0
GAL4	CGGNNNNNNNNNNCCG	CGGNNNNNNNNNNCCG	1	6,11
GCN4	TGANTN	NOT FOUND	-	6,6
GCR1	CWTCC	CTTCC	13	5,0
HAP1	CGGNNNTANCGG	GGANNNNCGG	1	6,6
HSE	TTCNNGAA	TTMTAGAA	6	8,0
	TTCNNGAA			
	GAANNNTCC			
	GAANNNTCC			
MCB	WCGCGW	ACGCGT	1	6,6
MCM1*	CCNNNWWRRGG	TTTCCTAA	1	8,0
MATa2	CRTGTWWWW	CATGTMA	2	7,0
MIG1	CCCCRNWWWWW	MCCCCAG	1	7,0
PHO4	CACGTK	CACGTG	1	6,6
PDR3	TCCGYGGA	TCCGYGGA	2	8,0
REB1	YYACCCG	YTACCCG	1	7,0
ROX1	YYNATTGTTY	CCTATTG	7	7,0
RAP1	RMACCCA	ACCCAGW	1	7,0
CAR1	AGCCGCSA	TAGCCGCS	2	8,0
SFF	GTMAACAA	NOT FOUND	-	-
STE12	ATGAAA	ATGNAAC	1	6,6
TBP	TATAWAW	NOT FOUND	-	-
UASPHR	CTTCCT	NOT FOUND	-	-
	GTSAAAGTAWG			

Table 1: Results on yeast regulons: Columns 1 and 2 are the regulon name and the known binding site consensus. The third column is the first close match to this consensus in the list of top 10 motifs reported by DMOTIFS. Column 4 is its rank in this list, and Column 5 gives the parameter values used: motif length, maximum number of spacers. In all experiments, a maximum of one degenerate symbol was allowed in the motifs being searched. * In the case of MCM1, the reported motif does not match the known consensus, but is found to occur as part of 12 of the MCM1 binding sites listed at SCPD. Hence it is counted as a match.

Sample size	t -score win	tie	TNoM p -value win
5	147	216	137
8	102	204	194
10	90	250	160
15	35	302	163
20	44	336	120

Table 2: Comparison of motif-finding performance of DMOTIFS using t -score and TNoM p -value score respectively. Column 1 indicates the number of input promoters (each of length 800) used in the experiment. Column 2 indicates how many times the t -score variant won (performed better), Column 3 counts the ties, and Column 4 counts the wins for the TNoM p -value variant.

Sample size	<i>RST</i> -score win	tie	TNoM <i>p</i> -value win
5	114	175	211
8	153	154	193
10	119	194	187
15	34	184	282
20	18	197	285

Table 3: Comparison of motif-finding performance of DMOTIFS using *RST*-score and TNoM *p*-value score respectively. Column 1 indicates the number of input promoters (each of length 800) used in the experiment. Column 2 indicates how many times the *RST*-score variant won (performed better), Column 3 counts the ties, and Column 4 counts the wins for the TNoM *p*-value variant.



Figure 1: Computation of the TNoM score: The gray boxes represent occurrences of the motif. For each positive promoter, a '+' is plotted, and for each negative promoter a '-' is plotted at the integer that is the count of the motif in that promoter. The best linear classifier is determined, and the number of errors made by it (3 in this case) is the TNoM score.

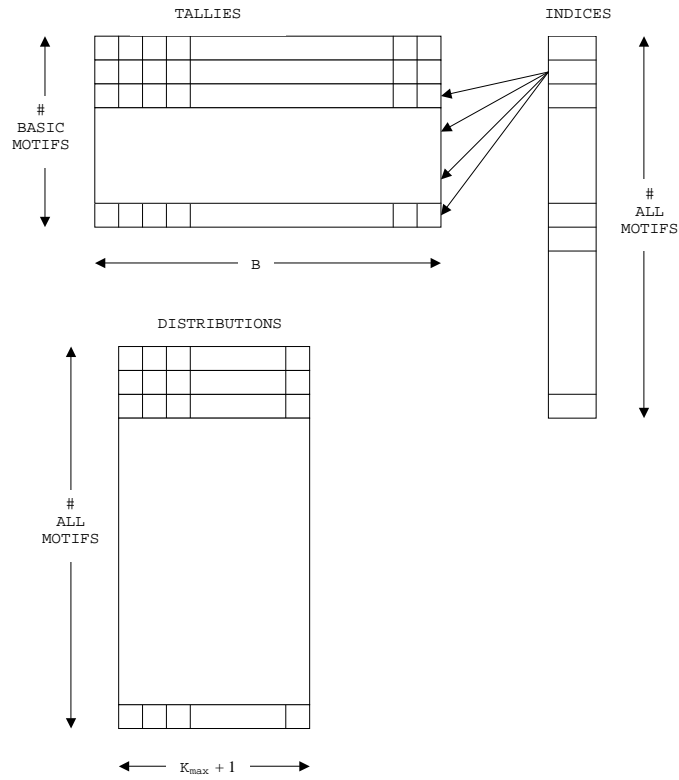


Figure 2: Data structures in simulations to estimate motif count distributions.