BIOINFORMATICS

Vol. 00 no. 0 2005, pages 1–5
doi:10.1093/bioinformatics/bti283

Finding Exact Optimal Motif in Matrix Representation by Partitioning

Henry C.M. Leung, Francis Y.L. Chin and Bethany M.Y. Chan

Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong

ABSTRACT

Motivation: Finding common patterns, or motifs, in the promoter regions of co-expressed genes is an important problem in bioinformatics. A common representation of the motif is by probability matrix or PSSM (Position Specific Scoring Matrix). However, even for a motif of length 6 or 7, there is no algorithm that can guarantee finding the exact optimal matrix from an infinite number of possible matrices.

Results: This paper introduces the first algorithm called EOMM for finding the exact optimal matrix-represented motif, or simply optimal motif. Based on branch-and-bound search by partitioning the solution space recursively, EOMM can find the optimal motif of size up to 8 or 9, and a motif of larger size with any desired accuracy on the principle that the smaller the error bound, the longer the running time. Experiments show that for some real and simulated data sets, EOMM finds the motif despite very weak signals when existing software, such as MEME and MITRA-PSSM, fails to do so.

Contact: {cmleung2, chin, mychan}@cs.hku.hk

1 INTRODUCTION

One important problem in bioinformatics is to understand how genes cooperate to perform functions, i.e. the gene regulatory network. Related to this is the sub-problem of *finding motifs* for co-regulatory genes.

The context behind the motif finding problem is the following. *Gene expression* is the process whereby a gene is decoded to form a RNA sequence which is then used to produce the corresponding protein sequence. In order to start the gene expression process, a molecule called a *transcription factor* will bind to a short substring in the promoter region of the gene. We call this substring a *binding site* of the transcription factor. A transcription factor can bind to several binding sites in the promoter regions of different genes to make these genes co-express, and such binding sites should have common pattern. The *motif finding problem* is to find the common pattern, or motif, from a set of promoter regions without knowing the positions of the binding sites.

Before we discuss how to find motifs, we need a model to represent motifs. There are two common models: string representation [Brazma et al., 1995; Buhler and Tompa, 2001; Chin et al., 2005; Leung and Chin, 2005; Li et al., 2002; Liang, 2003; Pevzner and Sze, 2000; Rocke and Tompa, 1998; Sagot, 1998; Staden, 1989; Tompa, 1999; Wolfertsteeter et al., 1996] and matrix representation [Bailey and Elkan, 1995; Barash et al., 2001; Chin et al., 2004; Eskin, 2004a; Lawrence et al.,1993; Leung et al., 2005; Liu et al., 1995].

String representation uses a length-l string of symbols (or nucleotides) A, C, G and T to describe a motif. In this model, the number of different length-l motifs is limited to 4^l and, when l is small, it is possible to find the "optimal" motif (highest-scoring with respect to some specified scoring function that compares the motif against binding sites). Unfortunately, many motifs in real biological data cannot be adequately described in this way. For example, the motif may be better described by allowing more than one symbol (e.g. not only G but G or T) to occupy a single position. Thus, some researchers have tried to improve string representation by introducing wildcard characters [Shinozaki et al., 2003; Sinha and Tompa, 2000] into the string to represent choice from a subset of symbols at a particular position. For example, K denotes G or T.

Matrix or PSSM (Position Specific Scoring Matrix) representation uses a $4 \times l$ matrix of real numbers to represent the motif where the j-th column of 4 numbers gives us the probability, respectively, that symbol A, C, G or T occupies the j-th position of the motif. Although the matrix is more expressive than the string, there are an infinite number of $4 \times l$ matrices of real numbers that form the solution space and no algorithm has been able to guarantee that the optimal motif, according to some scoring function, can be found. Thus, for matrix representation, there are computational hurdles to overcome.

This paper tackles the motif finding problem in which motifs are represented by matrices. Our solution extends the novel idea used in the algorithm MITRA-PSSM [Eskin, 2004a]. MITRA-PSSM partitions the infinite matrices in the solution space into a fixed number of categories based on strings of nucleotides {A, C, G, T} and wildcard characters representing two nucleotides. Then it estimates an upper bound on the scores of all matrices within each category. For categories that have high estimated scores, a local search, based on Expectation-Maximization (EM) theory, is performed to find the optimal matrix within each category. Unfortunately, MITRA-PSSM's "partitions" (categories) are not really partitions because they do not cover all matrices in the solution space and are not disjoint. Moreover, there is no guarantee that the EM algorithms will find the optimal solution within each category, and with incomplete coverage of the solution space, the optimal matrix may even not fall into a category to be considered.

Our contribution is to introduce an algorithm EOMM (i.e. Exact Optimal Motif in Matrix representation) to find the exact optimal scoring motif in matrix representation (calculating the exact values of all entries in the optimal matrix), or simply optimal

The research was supported in parts by the RGC grant HKU 7135/04E

Table 1. Representative Column Vectors and their Motivation

Class	Group	Motivating vector (MV) value set	Representative column vector (RCV) value set
Strictly conserved	A	(1.0, 0.0, 0.0, 0.0)	(1.0, 0.0, 0.0, 0.0)
	C	(0.0, 1.0, 0.0, 0.0)	(0.0, 1.0, 0.0, 0.0)
	G	(0.0, 0.0, 1.0, 0.0)	(0.0, 0.0, 1.0, 0.0)
	T	(0.0, 0.0, 0.0, 1.0)	(0.0, 0.0, 0.0, 1.0)
Strictly 2-symbol	M	(0.5, 0.5, 0.0, 0.0)	(0.75, 0.75, 0.00, 0.00)
	R	(0.5, 0.0, 0.5, 0.0)	(0.75, 0.00, 0.75, 0.00)
	W	(0.5, 0.0, 0.0, 0.5)	(0.75, 0.00, 0.00, 0.75)
	S	(0.0, 0.5, 0.5, 0.0)	(0.00, 0.75, 0.75, 0.00)
	Y	(0.0, 0.5, 0.0, 0.5)	(0.00, 0.75, 0.00, 0.75)
	K	(0.0, 0.0, 0.5, 0.5)	(0.00, 0.00, 0.75, 0.75)
Unconserved	N	(0.25, 0.25, 0.25, 0.25)	(0.375, 0.375, 0.375, 0.375)
Slightly conserved	a	(0.75, 0.08, 0.08, 0.08)	(1.00, 0.12, 0.12, 0.12)
	c	(0.08, 0.75, 0.08, 0.08)	(0.12, 1.00, 0.12, 0.12)
	g	(0.08, 0.08, 0.75, 0.08)	(0.12, 0.12, 1.00, 0.12)
	t	(0.08, 0.08, 0.08, 0.75)	(0.12, 0.12, 0.12, 1.00)
Slightly 2-symbol	m	(0.4, 0.4, 0.1, 0.1)	(0.60, 0.60, 0.15, 0.15)
	r	(0.4, 0.1, 0.4, 0.1)	(0.60, 0.15, 0.60, 0.15)
	w	(0.4, 0.1, 0.1, 0.4)	(0.60, 0.15, 0.15, 0.60)
	s	(0.1, 0.4, 0.4, 0.1)	(0.15, 0.60, 0.60, 0.15)
	y	(0.1, 0.4, 0.1, 0.4)	(0.15, 0.60, 0.15, 0.60)
	k	(0.1, 0.1, 0.4, 0.4)	(0.15, 0.15, 0.60, 0.60)

RCV is a vector which is $2^{0.58}$ times the corresponding MV with the restriction that each entry is at most 1.

motif, which, through a recursive partitioning of the solution space, has the features of robustness, efficiency and accuracy as explained in the next section.

This paper is organized as follows. Section 2 gives background on the dividing method used by MITRA-PSSM. Section 3 describes the details of matrix representation. In Section 4, we describe how to divide the infinite matrices into categories by dividing the column vectors of the matrices into partitions and how to calculate the upper bound on scores for each category/partition. Algorithm EOMM is described in Section 5. Experimental results on both real data and simulated data are shown in Section 6, followed by a discussion in Section 7.

2 BACKGROUND

MITRA-PSSM [Eskin, 2004a] finds motifs which are represented by matrices. First, the solution space of 4×l matrices is partitioned into 21^l categories which are represented by length-l strings taken from an alphabet of 21 symbols, resembling the IUPAC alphabets [Sinha and Tompa, 2002] which represent single nucleotides and combinations of two nucleotides. Then an upper bound on the score of a solution within each category is calculated. The next step is to search through the strings to identify categories which could contain the optimal solution using a branch-and-bound approach similar to SPELLER [Sagot, 1998], and then to search for the solution within each such category using EM algorithms.

The 21-character alphabet (Table 1) for the strings represents the "centers" of 21 groups of column vectors (of the motif matrix). Note that the 4×l matrices are partitioned into "categories" while column vectors into "groups" in MITRA-PSSM or "partitions" in our algorithm EOMM. Each "center" can be represented by a *motivating vector* (MV). Column vector (v_A,v_C,v_G,v_T) is in the group with MV (m_A,m_C,m_G,m_T) iff $\log(v_\alpha) \le \log(m_\alpha)+0.58$ or $v_\alpha \le 2^{0.58}m_\alpha \approx 1.5m_\alpha$ where α can be any nucleotide {A,C,G,T} and $v_A+v_C+v_G+v_T = m_A+m_C+m_G+m_T = 1$. In order to better characterize these 21 groups, we further represent each group by a *representative column vectors* (RCV), a four-tuple of real numbers (r_A,r_C,r_G,r_T) obtained by scaling up each MV by a factor of $2^{0.58} \approx 1.5$, more precisely $r_\alpha = \min\{2^{0.58}m_\alpha,1\}$ where $\alpha\in\{A, C, G, T\}$. In other words, an arbitrary column vector (v_A,v_C,v_G,v_T) is in the group with RCV (r_A,r_C,r_G,r_T) iff $0 \le v_\alpha \le r_\alpha$ where $\alpha\in\{A, C, G, T\}$. For example, column vector (0.1, 0.9, 0, 0) is in group c represented by RCV (0.12, 1, 0.12, 0.12). Similarly, column vector (0, 0, 0.4, 0.6) is in group K represented by RCV (0, 0, 0.75, 0.75) where K is the IUPAC alphabet for nucleotide G or T.

The 21 different MVs (or RCVs) are not defined arbitrarily, but have some motivation behind them. They are classified into 5 classes: Strictly conserved, Slightly conserved, Strictly 2-symbol, Slightly 2-symbol, and Unconserved. In order to map a 4×l matrix to the 21^l groups, each column of the matrix is mapped into the appropriate RCV which represents that column, resulting in a mapping of the matrix to a category as described by a length-l string with 21 alphabets. For example, when l = 5, a 4×5 matrix

$$M = \begin{pmatrix} 0.1 & 0 & 0.8 & 0.2 & 0.6 \\ 0.9 & 0 & 0.1 & 0.25 & 0.2 \\ 0 & 0.4 & 0.1 & 0.35 & 0.1 \\ 0 & 0.6 & 0 & 0.2 & 0.1 \end{pmatrix}$$

is in the category represented by "cKaNm" or "ckaNm"

Unfortunately, because of the way the 21 MVs or RCVs are defined (Table 1), some column vectors do not lie in any group (i.e. incomplete partitioning) and some lie in more than one group (groups are not disjoint). For example, column vector (0.4, 0.3, 0.2, 0.1) does not lie in any group, while the second column vector (0, 0, 0.4, 0.6) of the above matrix M (another example is (0, 0, 0.5, 0.5)) lies in groups K and k with corresponding RCVs (0, 0, 0.75, 0.75) and (0.15, 0.15, 0.6, 0.6) respectively. An important consequence of not covering the solution space completely is that the optimal solution may not be found. The consequence of non-distinct groups is that more groups may have to be considered, especially when the optimal or good solutions fall into more than one group.

As mentioned in the introduction, we focus on how to partition the set of the column vectors and introduce EOMM with the following features:

(1) [Robustness through true partitioning] EOMM performs a true partitioning of the solution space, giving disjoint categories that cover the entire solution space so that no (optimal) solution will be missed.

(2) [Efficiency with fewer categories] EOMM uses fewer categories (9^l instead of 21^l) than MITRA-PSSM to cover the entire solution space, without sacrifice to accuracy. Consequently, in EOMM, much fewer categories have to be considered when searching for the optimal solution. Moreover, disjoint categories are more efficient than non-disjoint categories, because, with non-disjoint categories, more categories may need to be studied when the optimal or near optimal solutions fall into more than one category. Note that the disjointedness and reduction in the number of categories can be achieved because the column vectors can be divided into 9 disjoint partitions instead of 21 non-disjoint groups.

(3) [Guaranteed Accuracy through recursion] EOMM's initial partitioning has 0.5 as the maximum error which is the same as that in MITRA-PSSM's. For example, a column vector (0.6, 0.1, 0.15, 0.15), which is in group m with RCV (0.6, 0.6, 0.15, 0.15), may have an error of 0.5 for nucleotide C. Moreover, EOMM has, as its parameter, error δ which controls the accuracy of the solution desired. Definitely, the smaller is the error δ, the longer the running time. Since all partitions in EOMM have the same properties in the sense that all column vectors in each partition can be characterized by four intervals of real numbers in the solution space, each partition can be further recursively divided into smaller partitions with the same properties. The error between the solution candidates and the optimal matrix can also be reduced accordingly until the solution matrix is within any desired accuracy. Finally when the partition is sufficiently small as well as the number of binding sites for solution candidates in the partition, a brute force approach will be employed to find the exact optimal matrix.

3 MOTIF MODEL

As in [Bailey and Elkan, 1995; Eskin, 2004a; Lawrence and Reilly, 1990], input sequences are broken up conceptually into a set of length-l overlapping substrings $\{s_i\}$. We assume that all these substrings are either generated according to an unknown probability matrix M or the background probability B. M is a 4×l matrix where $M(\alpha,j)$ is the probability that the j-th nucleotide of a length-l binding site is α and $M(\mathrm{A},j) + M(\mathrm{C},j) + M(\mathrm{G},j) + M(\mathrm{T},j) = 1$, $1 \leq j \leq l$. For any length-l substring σ, the probability that σ is generated by M is $M(\sigma) = \prod_{j=1}^{l} M(\sigma[j], j)$ where $\sigma[j]$ is the j-th nucleotide of σ. B is a 4-dimensional vector (B(A), B(C), B(G), B(T)) with $B(\mathrm{A}) + B(\mathrm{C}) + B(\mathrm{G}) + B(\mathrm{T}) = 1$, which represents the probability of each nucleotide occurring in the non-binding region. The probability that a length-l substring σ generated by B is $B(\sigma) = \prod_{j=1}^{l} B(\sigma[j])$. For each length-l substring s_i, we have a hidden variable a_i to indicate if s_i is a binding site. If the substring s_i is a binding site, $a_i = 1$; otherwise, $a_i = 0$. Given the value of the hidden variable a_i and the prior probability $P(M)$ of a substring being generated by M, the likelihood of the input sequences [Bailey and Elkan, 1995; Eskin, 2004a; Lawrence and Reilly, 1990] is

$$\begin{aligned} L &= \prod\nolimits_{s_i:a_i=1} P(M)M(s_i) \prod\nolimits_{s_i:a_i=0} \big(1 - P(M)\big) B(s_i) \\ &= \prod\nolimits_{s_i:a_i=1} \frac{P(M)M(s_i)}{\big(1 - P(M)\big)B(s_i)} \prod\nolimits_{s_i} \big(1 - P(M)\big) B(s_i) \end{aligned}$$

The motif finding problem is to find the probability matrix M and the hidden variables $\{a_i\}$ so as to maximize the log likelihood log(L) for the input sequences.

For a fixed M and P(M), the likelihood is maximized if $a_i = 1$ whenever $M(s_i)/B(s_i) \geq (1\text{-P}(M))/\text{P}(M)$, i.e., length-$l$ substring s_i is considered as a binding site. Then the maximum log likelihood is

$$\log(L) = \sum_{s_i:\log\left(M(s_i)/B(s_i)\right) > t} \left(\log\left(\frac{M(s_i)}{B(s_i)}\right) - t \right) + \sum_{s_i} \log\big(\big(1 - P(M)\big)B(s_i)\big)$$

where $t = \log\big(1 - P(M)/P(M)\big)$

Since $\sum_{s_i} \log\big(\big(1 - P(M)\big)B(s_i)\big)$ is independent of the probability matrix M, we define the *information content for a probability matrix M* as

$$IC(M) = \sum_{s_i:\log\left(M(s_i)/B(s_i)\right) > t} \left(\log\left(\frac{M(s_i)}{B(s_i)}\right) - t \right)$$

The motif-finding problem is now reduced to finding the probability matrix M so as to maximize IC(M). In [Eskin, 2004a], it is mentioned that flexibility exists because many types of additional information can be incorporated into this search problem by using different threshold value t, e.g. non-uniform background probability distribution, different prior probabilities of binding sites at different string positions, etc.

4 PARTITIONING OF THE SEARCH SPACE

Since there are infinite numbers of probability matrices M, we cannot find the optimal matrix with maximum IC(M) by exhaustion. In this section, we will describe how to divide the infinite number of probability matrices into categories and how to calculate the upper bound of IC(M) for all matrices M within a particular category. By further dividing the search space recursively, we can reduce the error between the solution matrix and the optimal matrix as small as needed.

4.1 Partition all Column Vectors

First we will describe how to divide all possible column vectors of any probability matrix M into partitions. Each entry of a column vector (v_A,v_C,v_G,v_T) can be any real number between 0 and 1 with the constraint that the sum of all entries in the column vector is 1. Figure 1 is the graphical representation of all possible column vectors. Since $v_\mathrm{A} + v_\mathrm{C} + v_\mathrm{G} + v_\mathrm{T} = 1$, we use a 3-tuple ($v_\mathrm{C}$,$v_\mathrm{G}$,$v_\mathrm{T}$) to represent a column vector (v_A,v_C,v_G,v_T). The x-axis, y-axis and z-axis represent the values of v_C, v_G and v_T respectively and the corresponding v_A of the column vector equals $1 - v_\mathrm{C} - v_\mathrm{G} - v_\mathrm{T}$. All column vectors can be mapped to a point lies in the tetrahedron with corner points (0,0,0), (1,0,0), (0,1,0), (0,0,1) which represent four column vectors (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1) respectively. Moreover, column vectors with the same value of v_A will lie on the same plane $v_\mathrm{C} + v_\mathrm{G} + v_\mathrm{T} = k$ where $k = 1 - v_\mathrm{A}$.

In order to have a partition whose maximum IC value can be estimated efficiently, we represent a partition P of column vectors by 4 ordered pairs (s_α, r_α) where $\alpha \in$ {A, C, G, T}. A column vector (v_A,v_C,v_G,v_T) is in partition P if $s_\alpha < v_\alpha \leq r_\alpha$ where $\alpha \in$ {A, C, G, T}, < is replaced by ≤ if $s_\alpha = 0$. Figure 2 gives the graphical representation of a partition. A partition is bounded by eight planes and its shape is like a cuboid with two corners removed. In order to

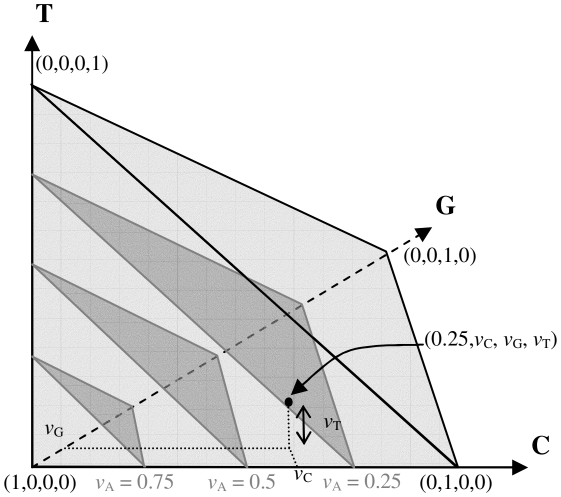

Fig 1. Graphical representation of all column vectors (v_A, v_C, v_G, v_T) of a probability matrix

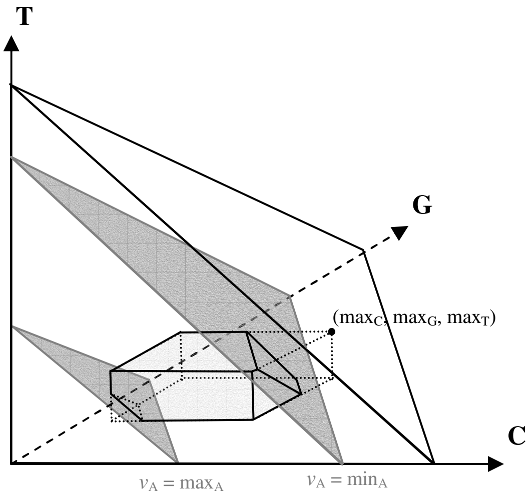

Fig 2. Graphical representation of a partition of column vectors

Table 2. 9 partitions of all possible column vectors

Class	Partition	((s_A, r_A], (s_C, r_C], (s_G, r_G], (s_T, r_T])
Strictly conserved	A	((0.85,1], [0,0.15], [0,0.15], [0,0.15])
	C	([0,0.15], (0.85,1], [0,0.15], [0,0.15])
	G	([0,0.15], [0,0.15], (0.85,1], [0,0.15])
	T	([0,0.15], [0,0.15], [0,0.15], (0.85,1])
Unconserved	N	([0,0.5], [0,0.5], [0,0.5], [0,0.5])
Slightly conserved	a	((0.5,0.85], [0,0.5], [0,0.5], [0,0.5])
	c	([0,0.5], (0.5,0.85], [0,0.5], [0,0.5])
	g	([0,0.5], [0,0.5], (0.5,0.85], [0,0.5])
	t	([0,0.5], [0,0.5], [0,0.5], (0.5,0.85])

make sure each column vector lies in exactly one partition, we need to divide the search space (the tetrahedron with corner points (0,0,0), (1,0,0), (0,1,0), (0,0,1)) into a set of non-overlapping partitions. The volume of a partition *P* over the volume of the tetrahedron (1/6 unit^3) represents the percentage of the column vectors which lie in the partition *P*. One might think that we should divide the search space into smaller tetrahedrons with equal shape and volume so that the probability that a randomly picked column vector would lie in each partition is equal. However, this partition method has two weaknesses. First, since we must bound the partition with 8 planes parallel to the planes $v_C = 0$, $v_G = 0$, $v_T = 0$ or $v_{C\,+}\, v_{G\,+}\, v_T = 1$, not all shapes can be described. For example, we cannot represent the tetrahedron with corner points at (0.3, 0, 0), (0, 0.3, 0), (0, 0, 0.3) and (0.2, 0.2, 0.2) using 4 intervals as some of the surfaces of this tetrahedron are not parallel to the planes $v_C = 0$, $v_G = 0$, $v_T = 0$ or $v_{C\,+}\, v_{G\,+}\, v_T = 1$. As a result, we cannot describe the small tetrahedron properly and will introduce overlapped partitions. Second, in practice, the probability of each column vector appearing in the motif is not equal. A motif usually contains many conserved columns. This means that the occurrence probability of a conserved column vector, with one entry significantly larger than the rest of the entries, in the motif is higher than the occurrence probabilities of the other column vectors. We need to divide the search space into partitions with different volumes. Those partitions near the four corner points (representing the conserved column vectors) should be smaller while the partition in the center (representing the unconserved column vectors) should be larger, so that the probability of a column vector being a solution in each partition is almost the same. Based on this idea, all column vectors are divided into 9 non-overlapped partitions {A, C, G, T, a, c, g, t, N} whose ordered pairs are shown in Table 2. Partitions A, C, G and T are closest to the four corner points, so their volume (0.00563 unit^3) is relatively small. On the other hand, the volume of partition N, near the center, is relatively large (0.08 unit^3).

It is easy to show that every column vector lies in exactly one of these partitions. Assume the *j*-th column of the optimal matrix M^* lies in partition *P*, the difference between each entry in the *j*-th column of M^* and the corresponding entry of any column vector in *P* is at most $\delta = \max\{ s_\alpha - r_\alpha| \ \alpha = A, C, G, T\} \leq 0.5$, the same maximum error as MITRA-PSSM's partitions. Therefore, if we replace the 21 groups used in MITRA-PSSM by the 9 partitions in Table 2, the running time of MITRA-PSSM will decrease without increasing the error bound. Moreover, our partition method has an advantage that we can further divide a partition into smaller partitions by planes parallel to planes $v_C = 0$, $v_G = 0$, $v_T = 0$ or $v_C + v_G + v_T = 1$ and determine in which partition the optimal matrix M^* should lie. For example, if we divide the partition A = ((0.85, 1), (0, 0.15), (0, 0.15), (0, 0.15)) by the plane $v_G = 0.075$, we will get two smaller partitions, ((0.85, 1), (0, 0.15), (0, 0.075), (0, 0.15)) and ((0.85, 1), (0, 0.15), (0.075, 0.15), (0, 0.15)). Note that it might not be necessary to partition *P* across all dimensions. Only the dimensions which introduce a large error δ will be partitioned.

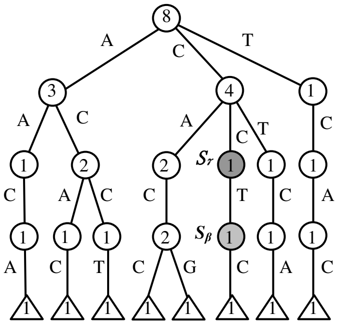

Fig. 3a

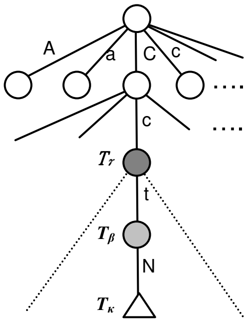

Fig. 3b

Fig. 3a. Suffix trie S (4-factor trie) for the sequence "AACACCTCACG" when $l = 4$. e.g. node S_γ represents the substring "CC" and its child node S_β represents the substring "CCT" where the prefix of "CCT" is "CC".
Fig. 3b. Category tree T when $l = 4$.

Thus, we can repeat this process with different numbers of partitions at each subsequent step so as to reduce the error δ.

4.2 Divide Matrix into Categories and Calculate the Upper Bound of its IC (Information Content)

Each column vector of a 4×l probability matrix M must lie in one of the 9 partitions of column vectors. Depending on which partition of each column vector of a probability matrix belongs to, we can divide all 4×l probability matrices into 9^l categories. Let M be a probability matrix in category W, represented by l partitions of column vectors from P_1 to P_l. Let M_W be the matrix formed by the maximum values of the corresponding partitions, i.e. $M_W(\alpha, j) = r_\alpha$ of the partition P_j, where $\alpha \in \{A, C, G, T\}$, $1 \leq j \leq l$. For example, matrix M in category CtaNa and its corresponding matrix M_W are

$$M = \begin{pmatrix} 0.1 & 0 & 0.8 & 0.2 & 0.6 \\ 0.9 & 0 & 0.1 & 0.25 & 0.2 \\ 0 & 0.4 & 0.1 & 0.35 & 0.1 \\ 0 & 0.6 & 0 & 0.2 & 0.1 \end{pmatrix} \quad M_W = \begin{pmatrix} 0.15 & 0.5 & 0.85 & 0.5 & 0.85 \\ 1 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.15 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.15 & 0.85 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

Note that the sum of all entries in the j-th column of M_W may larger than 1. Let the *binding sites of a probability matrix M* be those substrings s_i such that $M(s_i)/B(s_i) \geq (1 - P(M))/P(M)$. Similar to MITRA-PSSM [Eskin, 2004a], we can show in the Appendix that IC(M_W) is the upper bound of IC(M) for all matrices in category W and thus if IC(M_W) is small, then none of the matrices in category W can be the optimal matrix and W can be skipped.

Although MITRA-PSSM uses a similar method to find the motifs, its groups cannot cover all the probability matrices. For example, a probability matrix with a column vector (0.4, 0.3, 0.2, 0.1) is not in any of its groups. Moreover, MITRA-PSSM applies an EM-algorithm in each group, which tries, but cannot guarantee, the finding of the optimal matrix. The error of its solution, i.e., the difference between an entry of the solution matrix and the corresponding entry in the optimal matrix, can be as large as 0.5.

5 ALGORITHM

As SPELLER [Sagot, 1998] and MITRA-PSSM [Eskin, 2004a], we search among the categories by a branch-and-bound approach, but our search will continue by further partitioning the search space recursively or by exhausting all the possible solutions when the search space is sufficiently small. In this section, we shall describe the data structures used to support searching when the categories are further partitioned and how the number of possible solutions can be limited for brute-force searching. EOMM divides all possible 4×l probability matrices into 9^l categories. We use IC(M_W) as the upper bound of information content for all matrices in category W (see Appendix). If IC(M_W) is larger than the information content IC* of the best matrix found so far, we will further divide all matrices in category W into smaller categories and repeat the process until the error δ is less than a predefined threshold. However, since the number of categories to be tested increases exponentially with l, we use the following approach to speed up the computation of IC(M_W) and to reduce the number of categories to be tested.

5.1 Traverse a suffix trie

Since not all categories contain a probability matrix with high information content, the running time of the algorithm can be reduced if we do not consider those categories which would have a low IC(M_W). We use an approach similar to SPELLER and MITRA-PSSM to rule out those categories with low IC(M_W).

Two data structures, a suffix trie S and a category tree T, are maintained when searching for the optimal matrix. The suffix trie S represents all length-l' substrings with $0 \leq l' \leq l$ in the input sequences. This data structure is also called l-factor trie. Each length-l' substring s_γ in the input sequences can be represented by a particular node S_γ at level l'. Every length-(l'+1) substring s_β with prefix s_γ is represented by a child node S_β of S_γ. For each node S_γ, $f(S_\gamma)$ represents the number of occurrences of substring s_γ in the input sequences. Figure 3a shows a suffix trie for the sequence "AACACCTCACG".

The category tree T has a similar structure as suffix trie S. T represents all categories of 4×l' ($0 \leq l' \leq l$) probability matrices under consideration. Each category T_γ of 4×l' probability matrices is represented by a node at level l'. Each category T_β of 4×(l'+1) probability matrices with the first l' column vectors same as T_γ is represented as a child node of T_γ. Figure 3b shows a category tree. In the following, we describe the pruning condition for the branch-and-bound approach in finding the optimal IC* value. Let w be the maximum value of log($M(\alpha,j)$/B(α)) for any nucleotide α and position j. Assume leaf node T_κ is in the subtree rooted at the internal node T_γ, the information content of every 4×l probability matrix in category T_κ is at most

$$\mathrm{IC}_{\max}(T_\gamma) = \sum\nolimits_{s_{\gamma'} \in U} \left[\left(\log(M_{W\gamma}(s_{\gamma'}) / B(s_{\gamma'})) + (l - l')w - t \right) \bullet f(S_{\gamma'}) \right]$$

where U contains all substrings in the input sequences which may be binding sites of some matrix in T_κ, that is

$$U = \{ s_{\gamma'} \mid \log(M_{W_\gamma}(s_{\gamma'}) / B(s_{\gamma'})) + (l - l')w > t \}$$

and $S_{\gamma'}$ is the trie node representing the length-l' substring $s_{\gamma'}$. If IC$_{\max}$(T_γ) is less than the current maximum IC value, we need not consider all leaf nodes T_κ of the subtree rooted at T_γ and the subtree rooted at T_γ can be pruned. We calculate IC(M_{T_κ}) for all leaf nodes T_κ of the category tree T and update the current maximum IC value if IC(M_{T_κ}) is no less than the current maximum IC value.

Let {$S_{\gamma'}$} be the set of nodes in S such that log(M_{T_γ}($s_{\gamma'}$)/B($s_{\gamma'}$)) + (l - l')w > t for category T_γ of length l'. Consider a category T_β for 4×(l'+1) probability matrices with the first l' column vector same as T_γ. Let {$S_{\beta'}$} be the set of nodes in S such that log(M_{T_β}($s_{\beta'}$)/B($s_{\beta'}$)) + (l - l' - 1)w > t. $S_{\beta'}$ must be a child of node $S_{\gamma'}$ in S such that the prefix of $s_{\beta'}$ is $s_{\gamma'}$ and we can calculate IC$_{\max}$(T_β) based on IC$_{\max}$(T_γ) to reduce the running time as follows

$$\begin{aligned} & \log(M_{T_\beta}(s_{\beta'})/B(s_{\beta'}))+(l-l'-1)w \\ = \; & \log(M_{T_\tau}(s_{\tau'})/B(s_{\tau'}))+(l-l')w \\ & +\log(M_{T_\beta}(s_{\beta'}[l'+1],l'+1)/B(s_{\beta'}[l'+1]))-w \end{aligned}$$

5.2 Update data structure when dividing a category recursively

When we divide a category W into smaller categories {W_v}, we can simply construct a category tree T' for {W_v} and perform a depth-first search on T'. However, since each W_v is a partition of matrices in category W, the binding sites of M_W is a superset of the binding sites of all matrices in each W_v. Instead of searching binding sites in S, we construct a suffix trie S' for all binding sites of M_W, substring s_κ such that log(M_W(s_κ)/B(s_κ)) > t. Since the size of S' should be much smaller than S (suffix trie of all input sequences), the time needed for searching T' should be smaller than the time needed for searching T.

5.3 Derive the exact optimal matrix

When the number of patterns for the binding sites of M_W is small, that is, the number of different length-l substrings which are the binding sites of M_W is small, we use the brute force approach to find the optimal matrix M^* instead of dividing category W further.

Assume the binding sites of M_W have K different patterns {ρ_i}, and pattern ρ_i occurs k_i times in the input sequences. If the optimal matrix M^* is in category W, the binding sites of M^* must be a subset of these $\sum_i k_i$ binding sites. Assume the set of binding patterns of M^* is $\{\rho_i^*\}$, It is shown in [Eskin, 2004a] that M^*(α,j) = $(\sum_i k_i^* \bullet \mathrm{I}(\rho_i^*[j]=\alpha)/\sum_i k_i^*$ where I(p) returns 1 if the statement p is true and returns 0 otherwise. If K is small, it may be more efficient to find the corresponding M^* for each of the 2^K possible subsets of {ρ_i}, and update the optimal matrix with M^* with the maximum IC(M^*). If the optimal matrix lies in category W, we must be able to find the optimal solution M^* exactly with no error.

6 EXPERIMENTS

We implemented EOMM and tested it on both simulated and real biological data. All experiments were run on a P4 2.4G computer with 1GB memory, in which only 50MB memory was used.

6.1 Experiments on simulated data

We generated 10 length-500 DNA sequences with 0.25 as the occurrence probability of each nucleotide A, C, G and T, and planted 25 binding sites, according to a randomly-generated 4×7 probability matrix M, in the 10 DNA sequences at random positions. When we generated the data, the expected score E(M) of matrix M for each binding site was also calculated, where E(M) measures how easy the optimal motif can be found.

Table 3. Experimental results on simulated data

Expected score per binding site E(M)	Number of times the algorithm find the planted motif			
	MEME	MITRA-PSSM	EOMM	average time
-3.0 < E(M) ≤ -1.0	0 / 20	1 / 20	2 / 20	1.5 hour
-1.0 < E(M) ≤ 1.0	10 / 20	9 / 20	17 / 20	58 min
1.0 < E(M) ≤ 3.0	18 / 20	20 / 20	20 / 20	40 min

For each range of E(M), the experiment was repeated 20 times with different probability matrices and the number of successes for each algorithm was counted.

Table 4. Experimental results on real biological data

Transcription factor	Pattern of the published motif	Rank of the motif in the answer list		
		EOMM	MITRA-PSSM	MEME
ACE2	GCTGGT	2	-	-
BAS1	TGACTC	1	1	1
CuRE, MAC1	TTTGCTC	1	-	1
GATA	CTTATC	1	1	1
GCFAR	GGGCCC	1	1	1
GCRE, GCN4	TGANTN	1	1	1

The data are collected from the SCPD[22]. We show the pattern of the motif (instead of its matrix representation) to make it more readable. For each set of data, we look for motifs with length equal to the published motif. Rank is the position of the correct motif in the answer list. '-' means the algorithm cannot find the correct motif.

$$\begin{aligned} \mathrm{E}(M) &= \sum_j \left\{\sum_\alpha \left[M(\alpha,j)\bullet\log\left(M(\alpha,j)/B(\alpha)\right)\right]\right\}-t \\ &= \sum_j \left\{\sum_\alpha \left[M(\alpha,j)\bullet\left(\log M(\alpha,j)+2\right)\right]\right\}-t \\ &= \sum_j \left\{\sum_\alpha \left[M(\alpha,j)\bullet\log M(\alpha,j)\right]\right\}+\sum_j \left\{\sum_\alpha 2M(\alpha,j)\right\}-t \\ &= \sum_j \sum_\alpha \left[M(\alpha,j)\bullet\log M(\alpha,j)\right]+2l-t \\ &= 2l-\sum_j \left\{-\sum_\alpha \left[M(\alpha,j)\bullet\log M(\alpha,j)\right]\right\}-t \end{aligned}$$

which is 2l minus the sum of entropy of each column vector in M minus the threshold t. A high value for E(M) means that each binding site carries a strong signal of the motif and it is easier to find the motif [Chin et al., 2004, Leung et al, 2005].

We compared EOMM with two different algorithms, MITRA-PSSM and the popular motif-finding software MEME, for each set of simulated data. MITRA-PSSM finds the optimal matrix by partitioning the searching space into fixed categories and performing EM algorithm on those categories that may contain the optimal matrix. MEME finds the motif by using EM algorithm directly. Different random matrices M within each range of expected score E(M) were tested and the results are shown in Table 3. For each range of E(M), we repeated the experiment 20 times and counted the number of times the algorithms could find the correct motif. We say an algorithm can find the motif if matrix M is within the top 10 answers of the algorithm.

When the expected score for each binding site was large (1.0 < E(M) ≤ 3.0), all three algorithms found the correct motif most of the time. When the expected score decreased (-1.0 < E(M) ≤ 1.0), MITRA-PSSM and MEME might not be able to find the correct motif. This is because there might be many local maxima in the input sequences and EM algorithm does not guarantee that the optimal matrix can be found. Moreover, MITRA-PSSM failed to find the correct motif because it has been modified to improve its time complexity at the expense of accuracy [Eskin, 2004b]. However, with a longer execution time, EOMM could usually find the optimal matrix *M*. When the expected score decreased further (E(*M*) ≤ -3.0), no algorithm could find matrix *M* because the signal of matrix *M* was too weak and there were many matrices with information content larger than *M* [Chin et al., 2004, Leung et al, 2005].

6.2 Experiments on real biological data

SCPD[Zhu and Zhang, 1999] is a database of transcription factors for yeast. For each set of genes regulated by the same transcription factor, we chose the promoter regions of these genes as the input sequences. Table 4 shows the results of the three algorithms. On those real biological data with weak signal motif, EOMM works well when compared with MITRA-PSSM and MEME.

7 DISCUSSION

Most existing algorithms find matrix-represented motifs using local searching methods which do not guarantee that the optimal matrix can be found and the error of the solution can be very large. The MITRA-PSSM algorithm partitions the search space before performing the EM-algorithm. It can bound the error of the solution by 0.5 and has a higher probability of finding the optimal matrix than the other algorithms. In this paper, we introduce EOMM which divides the search space into fewer categories than MITRA-PSSM without increasing the error. Thus our algorithm should run faster than MITRA-PSSM before its modification. Moveover, EOMM can find motifs with any accuracy by partitioning the search space recursively.

Since EOMM usually takes much longer time to find the optimal motif, it is not advisable to use EOMM for discovering strong signal motif. However, EOMM outperforms all the existing algorithms to find motifs with very weak signal at the expense of long execution time. We can now find motif of length $l \leq 8$ in reasonable time, say a couple hours. For motifs with larger *l*, we can use local searching method to find all partitions containing at least one probability matrix with high score and then use EOMM to find the optimal matrix in these partitions.

Moreover, instead of using the maximum likelihood model when calculating the score of a matrix, we can extend our algorithm to use other models like maximum aposteriori (MAP) likelihood and Bayesian priors.

REFERENCES

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

Barash, Y., Bejerano, G. and Friedman, N. (2001) A Simple Hyper-Geometric Approach for Discovering Putative Transcription Factor Binding Sites. *WABI*, 278-293.

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

Buhler, J. and Tompa, M. (2001) Finding motifs using random projections. *RECOMB01*, 69-76.

Chin, F and Leung, H. (2005) Voting Algorithms for Discovering Long Motifs. *APBC*, 261-271.

Chin, F., Leung, H., Yau, S.M., Lam, T.W., Rosenfeld, R., Tsang, W.W., Smith, D. and Jiang, Y. (2004) Finding Motifs for Insufficient Number of Sequences with Strong Binding to Transcription Factor, *RECOMB04*, 125-132.

Eskin, E. (2004a) From profiles to patterns and back again: a branch and bound algorithm for finding near optimal motif profiles. *RECOMB04*, 115-124.

Eskin, E. (2004b) Personal communication.

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

Lawrence, C. and Reilly, A. (1990) An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function and Genetics*, **7**,41-51.

Leung, H., Chin, F., Yiu, S.M., Rosenfeld, R. and Tsang, W.W. (2005) Finding motifs with insufficient number of strong binding sites. *JCB* (to appear)

Leung, H. and Chin, F. (2005) Generalized Planted (*l*,*d*)-Motif Problem with Negative Set. *WABI* (to appear)

Li, M., Ma, B., and Wang, L. (2002) Finding similar regions in many strings. *Journal of Computer and System Sciences*, **65**, 73-96.

Liang, S. (2003) cWINNOWER Algorithm for Finding Fuzzy DNA Motifs. *Computer Society Bioinformatics Conference*, 260-265.

Liu, J.S., Neuwald, A.F., Lawrence, C.E.. (1995) Bayesian Motifs for Multiple Local Sequence Alignment and Gibbs Sampling Strategies. *Journal of the American Statistical Association*, **432**, 1156-1170.

Pevzner, P. and Sze, S.H.. (2000) Combinatorial approaches to finding subtle signals in dna sequences. *ISMB*, 269-278.

Rocke, E. and Tompa, M. (1998) An algorithm for finding novel gapped motifs in DNA sequences. *RECOMB98*, 228-233.

Sagot, M.F. (1998) Spelling approximate repeated or common motifs using a suffix tree. *In C.L. Lucchesi and A.V. Moura editors, Latin'98: Theoretical informatics, volume 1380 of Lecture Notes in Computer Science*, 111-127.

Shinozaki, D., Akutsu, T. and Maruyama, O. (2003) Finding optimal degenerate patterns in DNA sequences. *Bioinformatics*, **19(suppl. 2)**, ii206-ii214.

Sinha, S. and Tompa, M. (2000) A statistical method for finding transcription factor binding sites. *ISMB*, 344-354.

Sinha, S. and Tompa, M. (2002) Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, **30**, 5549-5560.

Staden, R. (1989) Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in Biosciences*, **5(4)**, 293-298.

Tompa, M. (1999) An exact method for finding short motifs in sequences with application to the ribosome binding site problem. *ISMB*, 262-271.

Wolfertsteeter, F., Frech, K., Herrmann, G., and Wernet, T. (1996) Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm. *Computer Applications in Bio-sciences*, **12(1)**, 71-80.

Zhu, J. and Zhang, M. (1999) SCPD: a promoter database of the yeast Saccharomyces cerevisiae. *Bioinformatics* **15**, 563-577. http://cgsigma.cshl.org/jian/

APPENDIX

Theorem: Let *M* be a matrix in category $W = P_1 P_2 \ldots P_l$. The set of binding sites of M_W is a superset of the set of binding sites of *M* and IC(*M*) ≤ IC(M_W).

Proof: Let ($s_{j,\alpha}$,$r_{j,\alpha}$) be the order pair in partition P_j represent the upper bound and lower bound of the occurrence probability of *α*. Since *M* is in category *W*, ∀ *α* = A, C, G, T and *j* = 1, …, *l*

$$
\begin{aligned}
& s_{j,\alpha} \leq M(\alpha, j) \leq r_{j,\alpha} \\
\Rightarrow\ & M(\alpha, j) \leq r_{j,\alpha} = M_W(\alpha, j) \\
\Rightarrow\ & \log\big(M(\sigma)/B(\sigma)\big) - t \leq \log\big(M_W(\sigma)/B(\sigma)\big) - t \\
& \qquad \text{for all length-}l\text{ string } \sigma \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1) \\
\Rightarrow\ & \log\big(M(\sigma)/B(\sigma)\big) - t > 0 \rightarrow \log\big(M_W(\sigma)/B(\sigma)\big) - t > 0 \\
\Rightarrow\ & \sum_{s_i : \log(M(s_i)/B(s_i)) - t > 0} \big(\log(M(s_i)/B(s_i)) - t\big) \\
& \leq \sum_{s_i : \log(M_W(s_i)/B(s_i)) - t > 0} \big(\log(M_W(s_i)/B(s_i)) - t\big) \\
\Rightarrow\ & \mathrm{IC}(M) \leq \mathrm{IC}(M_W)
\end{aligned}
$$

From the inequality (1) above, we can show that every binding site of *M* is also a binding site of M_W.