

**IDENTIFICATION OF TRANSCRIPTION FACTOR
BINDING SITES**

By

LIANG ZHAO

**Bachelor of Science
Zhejiang University
Hangzhou, China
1992**

**Master of Engineering
Beijing Research Institute of Chemical Industry
Beijing, China
1995**

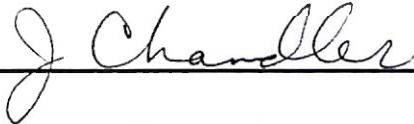
**Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the degree of
MASTER OF SCIENCE
May, 2004**

IDENTIFICATION OF TRANSCRIPTION FACTOR
BINDING SITES

Thesis Approved:



Thesis Advisor



Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my deepest appreciation to my major thesis advisor, Dr. H. K. Dai, for his invaluable help with my thesis, and his friendship. A rigorous scientist and an easy-going person, Dr. Dai made the thesis progress both inspiring and interesting. My deep appreciation also goes to my other committee members, Dr. George E. Hedrick and Dr. John P. Chandler. In addition to the help with my thesis work, Dr. Hedrick is the Chairperson of the Computer Science Department, and Dr. Chandler is the Director of the Department Graduate Programs during my study at Oklahoma State University. Always friendly and helpful, they are a blessing to me, an international student. Thanks.

I wish to thank Dr. Nohpill Park for his help. With Dr. Park, I took three classes and was a teaching assistant for three semesters. That was a joyful time.

I am also grateful to other faculty members, as well as the staff and graduate students in the Computer Science Department, who have helped me now and then.

I would like to thank my wife, Ms. Liqin Wang, and my parents, Mr. Yongde Zhao and Ms. Yulan Xiang. It is their love that helps me grow up and leads me to the happy life now.

Finally, I wish to give my sincere thanks to the Computer Science Department, Oklahoma State University for financially supporting me through most of the time of my study here.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. IDENTIFICATION OF DNA TRANSCRIPTION FACTOR BINDING SITES: A LITERATURE REVIEW.....	3
2.1 Introduction	3
2.2 Representations and applications	3
2.3 Extractions.....	8
3. PROGRESSIVE CONSENSUS PATTERN COUNTING: DETECTION OF CONSERVED PATTERNS AND IDENTIFICATION OF TRANSCRIPTION FACTOR BINDING SITES IN PROKARYOTES	14
3.1 Introduction	14
3.2 Materials and methods	17
3.3 Results and discussions.....	22
3.4 Conclusion and future work	26
4. GS_PC: A COMBINED APPROACH TO THE IDENTIFICATION OF TRANSCRIPTION FACTOR BINDING SITES IN PROKARYOTES.....	30
4.1 Introduction	30
4.2 Materials and methods	32
4.3 Results and discussions.....	36
4.4 Conclusion and future work.....	38
REFERENCES.....	41
APPENDIX A: THE THREE ESCHERICHIA COLI BINDING SITE SEQUENCE DATASETS.....	46
APPENDIX B: PART OF THE SOURCE CODE FOR PCPC AND GS_PC.....	59

LIST OF TABLES

2.1	An alignment matrix and the consensus binding site	4
2.2	PWM and information content.....	6
3.1	The three consensus binding site sequences used in this work.....	18
3.2	Three hypothetical pattern matching examples	19
3.3	Results from PCPC, based on expected frequency, as applied to the three binding site families in <i>E. coli</i>	23
3.4	Results from PCPC, based on z-score, as applied to the three binding site families in <i>E. coli</i>	24
3.5	Results from PCPC, based on total information content (TIC), as applied to the three binding site families in <i>E. coli</i>	25
4.1	GS_PC as run on the Stormo CRP sequence set.....	35
4.2	The best result at each cutoff value from Table 4.1.....	36
4.3	The best performance and best parameter setting of the old and new Gibbs sampler as run on the Stormo CRP dataset.....	39
4.4	The best performance and best parameter setting of the new Gibbs sampler as run on the LexA and purR datasets.....	39

LIST OF FIGURES

4.1	Flowchart of GS_PC.....	34
-----	-------------------------	----

NOMENCLATURE

bp	base pair
BS	binding site
CRP	cyclic AMP receptor protein
DNA	deoxyribonucleic acid
EM	expectation maximization
GS old	Gibbs sampler, the 1995 version
GS new	Gibbs sampler, the latest version as of April 12, 2004
PCPC	progressive consensus pattern counting
PSSM	position specific scoring matrix
PWM	position weight matrix
RNA	ribonucleic acid
TF	transcription factor
TFBS	transcription factor binding site

CHAPTER 1

INTRODUCTION

Understanding gene regulation is one of the major tasks in molecular biology, which has received a great boost since the recent development of deoxyribonucleic acid (DNA) microarray technology (Spellman *et al.*, 1998; Lashkari *et al.*, 1997; DeRisi *et al.*, 1997). Being able to monitor simultaneously the expression levels of virtually every single gene in an organism, DNA microarray technology allows the isolation of sets of co-regulated genes, i.e., genes that respond similarly to a perturbation in a defined cellular process.

The reason for the coordinated responses from a set of co-regulated genes is that these genes each have a similar segment in their DNA upstream regions. These segments can be recognized and bound to by the same type of DNA transcriptional factors (TFs), which, by at the same time interacting with a ribonucleic acid (RNA) polymerase, regulate the expressions of a particular set of genes.

These common DNA segments are called DNA transcriptional factor binding sites (TFBSs), or DNA regulatory sites. They are conserved, to various degrees, across an organism, and thus are statistically over-represented as compared to their surrounding background sequences, and can therefore be possibly identified by comparison of these regions. Identification of these DNA TFBSs is a fundamental step to the understanding of the gene regulation mechanism.

The rapid advancement in sequencing technique in the genome projects are producing explosively increasing DNA sequence data, totaling over several billions of symbols, which both necessitates and stimulates the utilization of high-performance computing in uncovering the biological information hidden in this vast quantity of sequence data.

This thesis focuses on the identification of DNA TFBSs. It is arranged as follows: Chapter 2 gives a brief literature review. Chapters 3 and 4 then propose two novel algorithms for this identification problem.

CHAPTER 2

IDENTIFICATION OF DNA TRANSCRIPTION FACTOR BINDING SITES: A LITERATURE REVIEW

2.1 Introduction

The problem of TFBS discovery has two facets: (1) representation and application, i.e., how to represent the specificity of a known or unknown family of TFBSs, and the procedures of applying the model to find new, *a priori* unknown sites, and (2) extraction, i.e., given a collection of DNA sequences that are known or expected to contain binding sites (BSs) for a common TF, how to locate the positions of these sites and extract the common patterns from them.

2.2 Representations and applications

There have been up to now generally two ways to represent the specificity of a TFBS family: consensus sequences, and position weight matrices (Stormo, 2000).

2.2.1 Consensus Sequences

Consensus sequences are the simplest and oldest methods to represent the specificity of a

Table 2.1 An alignment matrix and the consensus binding site: (a) the alignment matrix for a set of hypothetical binding sites, (b) the corresponding consensus binding site.

(a)	t g t g a c
	t c t g a g
	a g g g a c
	t g a a a c
	t g t g a t
	g g g g a c
(b)	t g t g a c

TFBS family (Maniatis *et al.*, 1975; Pribnow, 1975). Basically, a consensus sequence is a sequence that matches closely, but not necessarily exactly, to all example sites. More specifically, it is a sequence that has the same length as the example sites that it models, and each position of it is a symbol that occurs the most frequently in the corresponding column of all the example sites, i.e., a consensus. A consensus sequence may be defined either over the basic nucleotide alphabet $\Sigma = \{a, c, g, t\}$, or more generally, it can be written in the form of a regular expression, allowing wildcard symbols as well as spacer(s) of fixed or variable length (Mehldau *et al.*, 1993). Table 2.1 gives a simple consensus sequence example.

After a consensus sequence is obtained, classical pattern matching algorithms can then be applied to locate the positions in a test sequence where a pattern match is present.

Although the consensus sequence method has been widely used, it is now generally believed inadequate in modeling the specificity of a subtle TFBS family (Bucher *et al.*, 1996). There are many reasons. One is that each nucleotide position in a BS contributes differently to its binding activity; some positions are more important, indicated by their higher degrees of conservations; while others are less important and less conserved. A single pattern sequence can hardly model this variability well.

2.2.2 Weight matrices

A better and now more common representation of a TFBS family is a position weight matrix (Hertz and Stormo, 1996).

A position weight matrix (PWM) for a set of regulatory sites has four rows, each for one of the four possible nucleotides, and has the same number of columns as the number of positions in the binding sites that it models. The value of each element of the matrix w , $w(b,i)$, is called a weight. The weights of the matrix can be determined experimentally (Stormo *et al.*, 1986; Fields *et al.*, 1997), or they can be derived from the alignment matrix of the putative sites as in equation (2.1) (Staden, 1984; Hertz *et al.*, 1990; Tatusov *et al.*, 1994):

$$w(b,i) = f_{b,i} \log_2 \left(\frac{f_{b,i}}{p_b} \right), \quad (2.1)$$

Table 2.2 PWM and information content: (a) the PWM for the set of example sites in Table 2.1 (a), (b) the information content of the test sequence *tgtgac*.

(a)

	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
<i>a</i>	-0.58	-1.58	-0.58	-0.58	2.00	-1.58
<i>c</i>	-1.58	-0.58	-1.58	-1.58	-1.58	1.42
<i>g</i>	-0.58	1.74	0.42	1.74	-1.58	-0.58
<i>t</i>	1.42	-1.58	1.00	-1.58	-1.58	-0.58

(b)

$$\begin{aligned}
 I(tgtgac) &= 1.42 + 1.74 + 1.00 + 1.74 + 2.00 + 1.42 \\
 &= 7.29
 \end{aligned}$$

where $w(b,i)$ is the specific weight carried by nucleotide b when it occurs at position i in a putative site, $f_{b,i}$ is the observed frequency of nucleotide b at position i of the collection of putative sites, and p_b is the frequency of nucleotide b in the whole genome, or more appropriately, the frequency of nucleotide b in the given data set (Hertz and Stormo, 1999). Table 2.2 (a) is the corresponding PWM for the alignment matrix in Table 2.1 (a).

After a weight matrix is derived, it can then be used to score every candidate BS on a test sequence to measure how closely each candidate site matches the pattern represented by the matrix, and thus in what possibility it is a true BS. The score, $I(s)$, for a candidate site is simply the sum of all the matrix values for that site's sequence:

$$I(s) = \sum_{i=1}^l w(b, i) \quad (2.2)$$

$$I(s) = \sum_{i=1}^l f_{b,i} \log_2 \left(\frac{f_{b,i}}{p_b} \right) \quad (2.3)$$

where l is the length of the binding site. See Table 2.2 (b) for an example test sequence and its information content.

Stormo *et al.* (1982) find that the matrix representation is both more sensitive and more precise than the best available consensus sequence method in searching for new BSs. Mulligan *et al.* (1984) show that there is a strong correlation between the score for a particular DNA segment and its quantitative activity as a BS, thus demonstrating that a PWM is a reasonable representation of a TFBS family.

This simple weight matrix representation for a binding site implies that each position contributes independently to the site's total activity, which, though does not fit the experimental data perfectly, is in most cases a very good approximation of true TF - BS interactions (Benos *et al.*, 2002).

There have also been various improvements over this simple weight matrix concept. Durbin *et al.* (1998) use a pseudo-count to deal with the problem of a small learning set from which the weight matrix is to build. Frech *et al.* (1993) further take into account the fact that a training set may contain redundant information and thus eliminate closely related promoter sequences. Duret *et al.* (1997) suggest that each example sequence in the learning set be weighted by the binding strength to the transcriptional factor, but this information, though valuable, is rarely available.

2.3 Extractions

The methods for locating and extracting over-represented motifs (i.e., locally conserved regions, e.g., patterns) in the upstream regions of a set of co-regulated genes can be divided into two major categories: enumerative methods, and alignment methods. Each of these methods has its advantages and disadvantages, and has application fields for which it is better suited.

2.3.1 Enumerative methods

Enumerative methods, also known as exact methods or word counting methods, are among the earliest approaches for the analysis of DNA TFBSs (Hawley and McClure, 1983).

Enumerative methods are based on the frequency analysis of oligonucleotides (i.e., “words” in the genomic sequence text) of a certain length (or length range) in the given promoter sequences set (van Helden *et al.*, 1998, 2000; Brazma *et al.*, 1998). It is conceivable that if an oligonucleotide is found highly over-represented than expected from the overall promoter sequence or the whole genome composition, then it is very likely to correspond to a functional regulatory site (or part of it), although the converse may not be true (Duret and Bucher, 1997).

In the most straightforward implementation of this approach, first the candidate word space and the significance measure are defined, then all words from the word space are enumerated one by one. For instance, if the words are of length 3 in the nucleotide alphabet Σ , then the algorithm can simply enumerate all words *aaa*, *aac*, *aag*, ..., *ttt*,

count how many occurrences each word is present in the data set, calculate their significance, and finally output the words that have the highest significances.

This simplest approach was first applied in the early 1980's (Queen *et al.*, 1982; Waterman *et al.*, 1984) and recently in van Helden *et al.* (1998, 2000) and Sinha and Tompa (2002).

This straightforward enumeration approach can be easily extended to accommodate more complicated patterns. Smith *et al.* (1990) have discovered patterns with non-conserved spacers. They enumerate all possible patterns consisting of three conserved positions with constant spacings within a preset range, i.e., patterns of the type $a_1-x(d_1)-a_2-x(d_2)-a_3$, where a_1 , a_2 , and a_3 are characters from the basic alphabet, d_1 and d_2 are the numbers of wildcard characters in between. A straightforward extension of this method has been reported by Suyama *et al.* (1995), which permits the discovery of patterns containing flexible length wildcard spacers.

Enumerative approaches often simply deliver a list of over-represented oligonucleotides, but some methods provide the capability of automatic grouping of the resultant motifs into consensus patterns, and thus present the results in a small number of putative regulatory elements that can be examined more easily by experts.

Enumerative approaches are gaining more popularities since the arrival of complete genome data. The availability of whole genome sequence data allows the expected frequencies for each specific oligonucleotide to be more precisely estimated, and therefore better results can usually be obtained (e.g., van Helden *et al.*, 1998). Also, enumerative methods are rigorous and exhaustive approaches, and are therefore

guaranteed to find the most statistically significant motifs given a set of promoter sequences.

2.3.2 Alignment methods

Basically, given a set of DNA non-coding sequences known to contain in each of them one or more BSs for a TF, alignment methods proceed by building and comparing a large number of multiple local sequence alignments, each for a set of candidate BS subsequences, and finally delivering a weight matrix derived from the best alignment as the model of the TFBS family. In most cases, the criterion for the best alignment is the one with the maximum information content. Trying to form all the possible multiple sequence alignments from even for a moderate data set is computationally prohibitive, to search for an optimal alignment, the alignment algorithms thus use various circumventing strategies, as described in the following two subsections.

In general, alignment approaches are much more efficient than word counting methods for the detection of large motifs with higher internal variation, which is typical of prokaryotic regulatory sites. However, they are essentially heuristic methods, and are not exhaustive. Also, they can be attracted by local optima, thus risking to miss important regulatory features when they are masked by highly attractive local maximal.

Alignment methods can be further divided into two subcategories: progressive alignment methods, and simultaneous alignment methods.

2.3.2.1 Progressive alignment methods

As implied by its name, a progressive alignment method builds up a complete alignment of the putative sites by adding in new sequences one at each iteration, and outputs the best alignment in the end. The criterion for identifying the best alignment of potential sites at each step is usually to choose the one with the highest information content.

A straightforward implementation of this approach can be found in (Stormo and Hartzell, 1989; Hertz *et al.*, 1990, 1999). Although this algorithm serves a good illustration for a typical progressive alignment approach, it is not a very effective method. An obvious drawback of this algorithm is that it uses only a small subset of the data for the early sequences processed, and thus is easily misled, often producing results that depend strongly on the order with which the sequences are fed to the program.

2.3.2.2 Simultaneous alignment methods

Simultaneous alignment methods are essentially expectation-maximization (EM) approaches. Given a set of DNA promoter sequences, like progressive alignment methods, simultaneous alignment methods also aim to deliver a weight matrix as a model of the BS family; but unlike progressive alignment methods, simultaneous alignment methods start with a complete set of randomly chosen putative sites, one from each sequence in the data set. It then iterates between two steps (expectation and maximization) to identify an optimal alignment. As shown in Section 2.2, given a matrix (even it is a random one), one can calculate the scores for all potential BSs on a sequence.

Using those scores, one obtains a weighted alignment of all the possible sites. The alignment is then used to derive a new (and hopefully better) matrix representation for those sites. These two steps are repeated until the parameters that best explain the data are obtained, or until a fixed number of iterations are reached.

This is the basic procedure for an EM approach, and was originally used by Lawrence and Reilly (1990).

Uncarefully chosen initial data sets may lead EM to converge inappropriately, or to a local maximum, or not at all. This problem is in part solved by using a Gibbs sampler approach (Lawrence *et al.*, 1993; Neuwald *et al.*, 1995).

Gibbs sampler is basically a stochastic equivalence of the EM approach. In the Gibbs sampler algorithm, the maximization step is replaced by one that increases the likelihood of observing the data with a certain probability only. The chances of reaching the maximum increase with the number of iterations. The Gibbs sampler can detect shared motifs in protein or nucleic acid sequences. Also, since Gibbs sampler is a non-deterministic approach, it might deliver different motifs in different runs of the program, thus allowing the detection of multiple motifs from each sequence.

Bailey, Elkan, and Grundy have also developed an EM variation to solve the multiple sequence alignment problems (Bailey and Elkan, 1995; Grundy *et al.*, 1996), which is implemented in the MEME (Multiple EM for Motif Elicitation) package. Although more frequently used for extracting consensus motifs from a set of protein sequences, MEME may also be applied to DNA sequences effectively. The algorithm is basically an EM approach with more parameters to estimate, among them, the number and length of sites

as well as the number of sequences where each site is present. It thus allows for the simultaneous identification of multiple patterns from each sequence.

MEME and Gibbs sampler are two of the most popular algorithms for multiple sequence alignments.

CHAPTER 3

PROGRESSIVE CONSENSUS PATTERN COUNTING: DETECTION OF CONSERVED PATTERNS AND IDENTIFICATION OF TRANSCRIPTION FACTOR BINDING SITES IN PROKARYOTES

3.1 Introduction

Identification of DNA transcription factor binding sites is a fundamental first step towards the understanding of the machinery governing gene regulation and expression. Recent invention of DNA microarray technology has greatly accelerated this task, which can quickly classify sets of genes that appear to be coregulated. The upstream regions of the coregulated genes can be explored by various approaches to identify the set of coregulated promoter sites (see Rombauts *et al.*, 2003; Vanet *et al.*, 1999; Brazma *et al.*, 1998 for reviews).

The consensus sequence is a widely used representation for the specificity of a binding site family (Ghosh, 1993; Wingender *et al.*, 1996). A consensus has the same length as the binding sites it models. In its simplest form, a consensus sequence consists only of symbols from the nucleotides alphabet Σ , each position featuring the most dominant residue. It is widely believed and observed, however, that different positions contribute differently to the activity of the binding sites, indicated by the various conservations at various positions. To better capture this variability, a more general form

of consensus is generally desired that further allows inclusion of set symbols and wildcards.

3.1.1 Conserved positions and pattern extraction

For a motif to exert its functionality, certain residues at certain positions may be crucial, and thus are conserved against mutation over evolution. These conserved residues (or positions) can be exploited to extract motifs in related biosequences. Smith *et al.* (1990) develop such a method, where motifs are modeled as 3-amino acid patterns of the form $a_1 s_1 a_2 s_2 a_3$, where a_1, a_2, a_3 are conserved amino acids, and s_1, s_2 are intervening spacer regions. Despite the simplicity in its form, this method has been proven very successful in extracting motifs in related protein sequences (Henikoff and Henikoff, 1991). It is later drastically enhanced by Neuwald and Green (1994), in which a much larger pattern space is explored.

Besides these enumerative approaches, some alignment-based methods are also devised that make use of the idea of conserved positions in their algorithms, for example, the Gibbs sampler (Lawrence *et al.*, 1993; Neuwald *et al.*, 1995). A central idea that underlies the highly credited Gibbs sampler is “fragmentation” or “column sampling”, i.e., only $c \leq w$ (the motif width) information-richest columns are used in advancing an evolving alignment towards its optimum.

3.1.2 Pattern matching, together with conserved positions, in identifying motif instances

After a motif pattern is established, it can be used to search for new, *a priori* unknown instances of the motif in new sequences. The usage of such patterns, however, is not always straightforward. A pattern that is not defined adequately or in insufficient accuracy will probably invoke a large number of random matches, whereas an over-fitting pattern is likely to miss many remote members of the family. Once again, conserved positions in the motif can be called for help. Fernandez de Henestrosa *et al.* (2000), in an effort to discover new LexA-regulated genes in *E. coli*, use five variants of the known LexA consensus binding sequence to examine the entire *E. coli* genome. Four of these patterns are intentionally weak, but each also have seven conserved positions. This approach, on one hand, allows them to perform a sensitive pattern search to locate all previously reported genes, as well as a number of new genes that are potentially LexA-regulated; on the other hand, it confines the pattern search to only certain sites, thus avoiding overwhelming random matches.

3.1.3 Progressive Consensus Pattern Counting: detection of conserved patterns and identification of transcription factor binding sites in prokaryotes

We present here a simple method “Progressive Consensus Pattern Counting” (PCPC) that targets on the specificity of pattern searches. Given a consensus pattern for a family of DNA regulatory sites and a corresponding collection of promoter sequences suspected of

being coregulated, our method effectively detects the most significant k -patterns (see section 3.2.3 for definition of the k -patterns) for a range of k values, based on comparison of expected frequency of the patterns (Hertz and Stormo, 1999). The matching segments usually represent true binding sites, and thus can be used to build a PWM that can then be used in place of the consensus pattern to locate regulatory elements in the sequence set, with improved specificity and sensitivity (Stormo, 2000). Our method thus introduces a novel and effective scheme for obtaining a PWM representation of a TFBS family, starting from a consensus pattern representation. We illustrate the method by applying it to three *E. coli* binding site families, those of cyclic AMP receptor protein (CRP), LexA and purR.

3.2 Materials and methods

3.2.1 Materials

The *E. coli* genes that are regulated by CRP, LexA and purR, respectively, are determined according to the compilations in McCue *et al.* (2002). The corresponding three sets of DNA upstream promoter sequences are then collected from the *E. coli* whole genome data archived in the GenBank database, Accession No. NC_000913. Each promoter sequence corresponds to the whole span between the transcription start of the regulated gene and the end of next upstream gene.

The three consensus patterns used in this work are listed in Table 3.1. They are determined by examination of the known binding sites compiled in DPInteract (Robinson

Table 3.1. The three consensus binding site sequences used in this work.

Transcription factor	Consensus pattern
CRP	AAATGTGA TCACATTT
LexA	TACTGTATATATATACAGTA
purR ACGCAAACGTTTGCCT

‘.’ denotes the wildcard.

et al., 1998), as well as results reported in (Stormo and Hartzell, 1989; Berg and von Hippel, 1988; de Crombrughe *et al.*, 1984) for CRP, (Fernandez de Henestrosa *et al.*, 2000; Lewis *et al.*, 1994) for LexA. Note that the consensus pattern is currently required to be in its simplest form, each position being either a single nucleotide symbol or the wildcard.

3.2.2 Statistical significance of patterns

When enumerative approaches are employed to search for biologically functional motifs in sets of related biosequences, it is generally assumed that the statistically most significant patterns in the pre-specified pattern space are the best candidates.

Tompa (1999) suggests that a good measure for estimating the statistical significance of a pattern is one that takes into account the background residue distribution (criterion *A*) and the absolute number of pattern occurrences (criterion *B*). This will suffice for some

Table 3.2 Three hypothetical pattern matching examples. (a): approximate pattern matching, allowing up to one mismatch. (b), (c): exact pattern matching.

(a)	Pattern	<i>AAA</i>	<i>TTT</i>	(c)	Pattern	<i>AAAA</i>	<i>TTTT</i>
	Occurrences	<i>aac</i>	<i>ttt</i>		Occurrences	<i>aaaa</i>	<i>tttt</i>
		<i>aga</i>	<i>ttt</i>			<i>aaaa</i>	<i>tttt</i>
		<i>taa</i>	<i>ttt</i>			<i>aaaa</i>	<i>tttt</i>
<hr/>				<hr/>			
(b)	Pattern	<i>AA..A</i>	<i>TT..T</i>				
	Occurrences	<i>aaaca</i>	<i>ttttt</i>				
		<i>aacga</i>	<i>ttttt</i>				
		<i>aagta</i>	<i>ttttt</i>				
		<i>aataa</i>	<i>ttttt</i>				
<hr/>							

situations. There are other circumstances, however, where the two criteria alone may be necessary, but insufficient. To understand why, let us take a look at several simple examples, as shown in Table 3.2.

First consider the example in Table 3.2 (a): *AAA* and *TTT* are candidate patterns, and approximate pattern matching approach is performed, allowing up to one mismatch. Suppose that the residue background frequencies are $f_A = f_T$. Since both patterns recruit the same number of sites, one has to conclude that these two patterns are of equal significance if the measure of significance is based only on criteria *A* and *B*. However, apparently pattern *TTT* is (much) more significant. The situation in the second example, Table 3.2 (b), is essentially the same.

The conclusion to be drawn from the above examples is that, when estimating the statistical significance of patterns, in addition to the two criteria of the background

residue distribution and the absolute number of pattern occurrences, a third criterion has to be considered, i.e., the homology among the pattern instances over the entire pattern length. This additional criterion brings the valuable information about the quality of individual pattern occurrences into the significance measure, and is in many cases highly desired. A statistical measure that is based only on criteria *A* and *B* will work fine when exact “word” counting is performed, as the example in Table 3.2 (c). Here a word is a pattern that has a single-residue symbol at each position, without allowing either multi-residue set symbols or wildcards. In this case, the homology measure is the same among all patterns, and can be safely dropped.

Both measures of significance in Smith *et al.* (1990) and in Neuwald and Green (1994) are problematic. In particular, the former measure does not include the absolute number of pattern occurrences, while the latter missed the homology consideration.

A measure that successfully accounts for all three criteria and is used in this work is the expected frequency of patterns as developed by Hertz and Stormo (1999). Given an alignment formed by the matching segments for a specified pattern, the expected frequency of the alignment estimates the expected number of times of random alignments having equal or greater information content simply by chance.

For comparison, we also evaluated the performance of two other measures in determining the most statistically significant patterns: *z*-score and the total information content of the site alignment formed by the pattern hits.

The *z*-score is calculated as follows (Alder and Roessler, 1972):

$$z = \frac{N_s - N \cdot p}{\sigma}, \quad (3.1)$$

where N_s is the actual number of occurrences of the pattern, N is the total number of possible pattern occurrences, p is the probability of the pattern occurring by chance, and σ is the standard deviation of the pattern occurrences, calculated as follows:

$$\sigma = \sqrt{N \cdot p \cdot (1 - p)}. \quad (3.2)$$

The calculation of σ and z assumes the simplest null model, where nucleotides are uniformly and independently distributed in the dataset.

The total information content of the alignment for the pattern is calculated as follows:

$$TI = I \cdot N_{seq}, \quad (3.3)$$

where N_{seq} is the number of sites in the site alignment for the pattern, and I is the information content calculated as follows (Schneider *et al.*, 1986):

$$I = \sum_{j=1}^w \sum_{b=A}^T f_{j,b} \cdot \log_2 \left(\frac{f_{j,b}}{p_b} \right), \quad (3.4)$$

where $j = 1 \sim w$ is the column number in the alignment, $b = A \sim T$ is the nucleotide at a column, $f_{j,b}$ is the frequency of nucleotide b at column j , and p_b is the frequency of nucleotide b in the whole dataset.

3.2.3 Algorithm summary

The program is implemented in ANSI C. It takes as input the name of the data file that should have the following format: (1) the consensus pattern, and (2) the promoter

sequences in FASTA format (Pearson and Lipman, 1988). In this study, a k -pattern is simply the consensus pattern with all its w but k nucleotide positions replaced by the wildcard, where w is the number of nucleotide positions in the consensus. For each k value, there are thus $\binom{w}{k}$ different k -patterns.

The program first reads in the consensus pattern and the sequence data. For each k -pattern of a particular k value, the algorithm then performs a simple exact pattern matching on both strands of the sequences. Overlapping occurrences are allowed, but duplicated sites are considered only once. The expected frequency of the k -pattern is then computed. All patterns of the same k value are compared. The most significant k -pattern is reported, together with the site alignment of the matching segments.

These steps are repeated for the specified range of k values, and for all three binding site families.

3.3 Results and discussions

The program implemented as described in Materials and Methods is run on the three sets of promoter sequences. The value of k is varied between 8 ~ 6, and the results from the usage of the expected frequency as the statistical measure are shown in Table 3.3.

Different motifs are differentially conserved in their instances. At the same k value, more conserved motifs are expected to generate more pattern hits than less conserved ones. An inspection of the known binding sites reveals that the conservation of these three regulatory families decreases in the order of LexA > purR > CRP. In agreement

Table 3.3 Results from PCPC, based on expected frequency, as applied to the three binding site families in *E. coli*.

Transcription factor	k	k -pattern	Expected frequency	# matched sites ^a	# true sites ^b	Ratio ^c
CRP	8GTG.....TCACA...	9×10^{-21}	15	14	1.2×10^{-3}
	7G.G.....TCACA...	3×10^{-29}	20 (15)	18	1.6×10^{-3}
	6GTG.....T..CA...	1×10^{-37}	37 (18)	26	3.0×10^{-3}
LexA	8	..CTG.AT..A....CA...	1×10^{-48}	16	16	6.5×10^{-3}
	7	...TG.....AT.CAG..	2×10^{-51}	17 (16)	17	6.9×10^{-3}
	6	...TG.....T.CAG..	8×10^{-52}	19 (17)	17	7.7×10^{-3}
purR	8G..AACG..T.C.T.	6×10^{-35}	13	13	3.1×10^{-3}
	7CG.AA.CG....C...	6×10^{-38}	14 (13)	14	3.3×10^{-3}
	6CG.AA..G....C...	6×10^{-38}	14 (14)	14	3.3×10^{-3}

a. number in the parenthesis is the number of sites also matched by the $(k+1)$ -pattern.

b. # true (experimentally determined) binding sites in the three upstream sequence sets: CRP, 61; LexA, 18; purR, 16.

c. ratio = # matched sites / total possible sites in the sequence set.

Table 3.4 Results from PCPC, based on z-score, as applied to the three binding site families in *E. coli*.

Transcription factor	<i>k</i>	<i>k</i> -pattern	z-score	# matched sites (false)
CRP	8	...TGTGA.....TC.C....	38.91	16 (1)
	7	...TG.GA.....CAC....	28.64	22 (5)
	6GTG.....T..CA...	20.70	37 (11)
LexA	8	..CTG.A.A.....CAG..	89.49	16 (2)
	7	..CTG.A.....CAG..	54.94	18 (2)
	6	..CTG.....CAG..	31.37	19 (3)
purR	8G.AA.CG..TGC.....	61.87	13 (0)
	7GCAA.CG....C.....	40.52	16 (2)
	6G..A.CG...GC.....	23.92	18 (4)

Table 3.5 Results from PCPC, based on total information content (TIC), as applied to the three binding site families in *E. coli*.

Transcription factor	<i>k</i>	<i>k</i> -pattern	TIC	# matched sites (false)
CRP	8	...TGT.A.....TCAC....	266.77	18(2)
	7G.GA.....T.ACA...	338.60	25(5)
	6GTG.....T..CA...	445.46	37(11)
LexA	8	..CTG.AT..A....CA...	281.40	16(0)
	7	..CTG.....TCAG..	295.22	18(2)
	6	...TG.....TCAG..	308.94	19(2)
purR	8G..AACG..T.C.T.....	262.69	13(0)
	7G.AAACG.....T.....	278.79	15(2)
	6CG..A.CG..T.....	299.03	18(5)

with this observation, the percentage of pattern matches (as measure by the ratio of the number of pattern matches versus the total number of possible sites) decreases in the order of LexA > purR > CRP at almost all k values.

For all three sequence sets, the specificity remains very high, $\geq 90\%$, for sufficiently large k value, $k \geq 7$. As expected, the simple k -pattern counting is not exhaustive. For example, at $k = 6$, when all three sequences sets begin to produce random matches, the coverage of the pattern matching is still less than 100%.

Note that the matches found at higher k values are usually a subset of those at lower k values. This is not surprising, since the most conserved k -pattern generally utilizes the first k most conserved positions within the consensus pattern. As a result, lower k -pattern is usually a “subpattern” of higher k -patterns.

As expected, the results from the z -score and total information content calculations are not as good as those from expected frequency, as seen in Tables 3.4 and 3.5. This difference is especially notable for the purR dataset. While the expected frequency method produces 13(0), 14(0), 14(0) pattern hits at k values of 6, 7, 8, respectively, the results from the z -score and total information content methods are 13(0), 16(2), 18(4) and 13(0), 15(2), 18(5), respectively, where the number in parentheses is the number of false positive pattern hits.

3.4 Conclusion and future work

We present a simple yet effective method that, given a consensus pattern for a family of transcription factor binding sites and a collection of promoter sequences believed to be coregulated, identifies the most significant k -patterns within the consensus for a range of

k values, based on comparison of expected frequency of the patterns. The sites matched by the longest k -pattern are usually a subset of true binding sites. While lower k -patterns have lower discriminating powers, the specificity of the pattern matches remains high for sufficiently large k . The sites found can be used to build a position weight matrix for searching the whole set of binding sites with improved precision. The method is illustrated by the application to three *Escherichia coli* regulons, CRP, LexA and purR.

We introduce here a simple idea of progressive consensus pattern counting. The algorithm effectively finds the most significant k -patterns within a consensus for a range of k values. In contrast to most current pattern matching methods that focus on the exhaustiveness of the pattern searching, our method is intentionally selective. For sufficiently large k values, such as $k \geq 7$, the patterns matches are usually highly specific.

The idea of PCPC is not an end in itself. The motifs instances found could be taken advantage of to build a PWM to facilitate the identification of the whole set of motif instances. To this end, it is not necessary to perform the pattern counting with a wide range of k values. Instead, 8 could be set as the upper bound for k , and 6 the lower bound. This should suffice for most cases. In case extremely conserved or degenerate motifs are to be handled, the upper bound can be raised or the lower bound be lowered slightly.

Matching segments from the highest k -pattern should be used first during the construction of the PWM, since the specificity of the matched sites is the highest. If it is decided that the number of matches are not enough, lower k -pattern hits can be used progressively. Throw away apparent random matches along the way of construction, and stop as soon as enough sites are collected.

As another possible usage, this idea of PCPC can be built into exhaustive pattern matching programs, for example those of Wishart *et al.* (1994) and Mehldau and Myers (1993), to render a more automated and more precise searches. Until now, users of such programs frequently have to experiment a large number of combinations of parameters to reach the most appropriate program settings. Incorporation of this idea could allow the user to quickly and reliably gather a subset of true binding sites. The PWM constructed can then be used to filter the exhaustive pattern matches. This would be especially helpful when “noise” sequences (those that do not contain a motif instance) constitute a considerable portion of the sequence data. In such circumstances, a simple powerful pattern match will probably incur a large number of false positive hits, most of which can be readily removed by scoring against the pre-constructed PWM.

The consensus pattern used in this work is in a simple form, consisting only of nucleotide symbols and wildcards. This risks the possibility that the statistically most significant k -patterns might escape examination. But this is in reality not a problem. The method of PCPC is not intended to be exhaustive in either sense: the whole set of motif instances may not be discovered, nor are the statistically most significant k -patterns in the whole sequence data guaranteed to be found. We are mainly concerned that a sufficiently significant k -pattern be found at each k value that can provide us a subset of true binding sites.

In this work, PCPC is illustrated on the identification of transcription factor binding sites in *E. coli*. But this should not be the limit for it. We expect that it will work equally well in other similar situations.

Three statistical measures are compared in selecting the most statistically significant patterns: expected frequency, z-score and total information content. The expected frequency measure obviously outperforms the other two measures in our study. This is not surprising. To qualify for a good statistically significant measure, three criteria need to be considered: background nucleotide distribution, total number of actual occurrences, and the homology among the pattern occurrences. The z-score lacks the consideration of the homology criterion, and is hence only suited for the calculation of exact “word counting”, while this is not the case in this study: we are doing pattern counting. For the total information content measure, at a first glance, it seems a bit surprising that it is not performing well, since it does incorporate all three criteria. A possible explanation is that it is a brute force consideration of the second criterion of total number of occurrences, simply multiplying it with the information content values, and thus lacking sophisticated weighing scheme, as done in the case of expected frequency developed by Hertz and Stormo (1999).

CHAPTER 4

GS_PC: A COMBINED APPROACH TO THE IDENTIFICATION OF TRANSCRIPTION FACTOR BINDING SITES IN PROKARYOTES

4.1 Introduction

4.1.1 Enumerative approaches and TFBS in prokaryotes

Traditionally enumerative approaches are mostly used in the identification of TFBSs in eukaryotes, such as the yeast *Saccharomyces cerevisiae* (van helden *et al.*, 1998, 2000; Sinha and Tompa, 2002), where short, more conserved sites are typical. For prokaryotes, such as *E. coli*, the problem is usually solved via alignment-based methods, for example, Gibbs sampler (Lawrence *et al.*, 1993; Neuwald *et al.*, 1995), CONSENSUS (Hertz and Stormo, 1999), and MEME (Bailey and Elkan, 1995; Grundy *et al.*, 1996). The reason is that the TFBSs in prokaryotes are usually much longer than their counterparts in eukaryotes, typically 20 bp long (Schneider *et al.*, 1986). A simple calculation makes clear why the usual from-the-scratch, brute force enumerative approach is not appropriate: To somewhat guarantee that the search is exhaustive, one has to inspect a pattern space of size $\binom{25}{6} \cdot 4^6 \approx 1 \times 10^9$, assuming only 6-patterns are to be examined.

Even for a moderate dataset, the time requirement will be prohibitive.

Various techniques have been designed to reduce the candidate pattern space with various successes (Neuwald and Green, 1994; Sagot and Viari, 1996; Jonassen *et al.*, 1995, 1997).

4.1.2 Combined approach in the identification of TFBSs

An obvious way of a combined approach is to first use an enumerative approach for a systematic scan of all statistically significant patterns in the example sequences, and then, based on the spotted significant patterns, initialize an alignment matrix.

The most straightforward implementation of this approach is first used in the late 1980's (Staden, 1989; Smith and Smith, 1990). After the alignment is obtained, Staden (1989) further calculates its information content to obtain a ranking of those discovered patterns with regard to their statistical significances, while Smith *et al.* (1990) extract a diagnostic pattern from the alignment, which can be used to test a new sequence to determine if it belongs to the family represented by the pattern.

Wolfertstetter *et al.* (1996) described a similar approach as that of Smith *et al.* (1990). They first identify patterns of a certain length that occur in a minimum percentage of all the sequences with certain mismatches. By hypothesizing that, in contrast to true sites, random patterns have no preferred mismatch positions, they thus can be eliminated. The remaining patterns are then extended laterally by incorporating the flanking conserved regions.

4.1.3 GS_PC: a combined approach to the identification of TFBSs in prokaryotes

We present here a novel combined approach to addressing this problem: “Gibbs sampler – Pattern Counting” (GS_PC). Specifically, we develop a combined approach that first utilizes a reliable alignment-based algorithm, for example, the Gibbs sampler (Lawrence *et al.*, 93; Neuwald *et al.*, 95), to locate some strong sites from the given dataset. These sites are most likely an inexhaustive collection of all true BSs present in the data; however, they offer a reliable consensus pattern that can be used as a significantly reduced pattern space, and thus drastically reduce the time required to perform the pattern search. For example, for the same site length of 25 bp, we now need only to check

$\binom{25}{6} \approx 2 \times 10^5$ patterns. When coupled further with the powerful, innovative

‘fragmentation’ technique which is an integral part of the Gibbs sampler, we can further

reduce the pattern space to mere several thousands $\binom{15}{6} \approx 5 \times 10^3$.

As revealed in Chapter 3, the expected frequency measure (Hertz and Stormo, 1999) is an effective measure for selecting the most statistically significant patterns, we therefore use it in this Chapter.

4.2 Materials and methods

4.2.1 Materials

A sequence dataset for the CRP protein has been proven useful in testing several popular algorithms (Stormo and Hartzell, 1989; Lawrence and Reilly, 1990; Hertz and Stormo,

1999), and is used here as the training set for our GS_PC algorithm (hereafter referred to as Stormo CRP sequence dataset). The same datasets of LexA and purR as used in Chapter 3 are then used to test the performance of our algorithm.

4.2.2 Algorithm summary

Gibbs Site Sampler, with fragmentation enabled, is used first on the dataset. Each segment located is filtered by comparing its information content (I) to the highest information content (HI) of all the segments; those with $\log(I)/\log(HI) < \text{cutoff}$ is removed from site alignments. Here the cutoff is a user-defined heuristic cutoff value; segments with information content higher than this value are pre-assumed to be true BSs, while those with information content lower than it are regarded as random background matches, and are weeded out.

After each site segment is thus examined, the sites left in the alignment are used to derive a simplified consensus pattern. All 6-patterns obtainable from this consensus are then enumerated, one after one, allowing up to one mismatch. Overlapping sites are not allowed, according to the biological motif model (Hertz and Stormo, 1999). All sites matched for a 6-pattern are aligned together, and the information content of each site segment is calculated based on the whole alignment excluding itself. Then the filtering process is performed again following exactly the same procedure: each site segment with information content ratio less than the user-specified cutoff is removed. Finally the expected frequency of the site alignment formed by all the remaining sites is calculated.

Figure 4.1 illustrates the flowchart of the algorithm GS_PC.

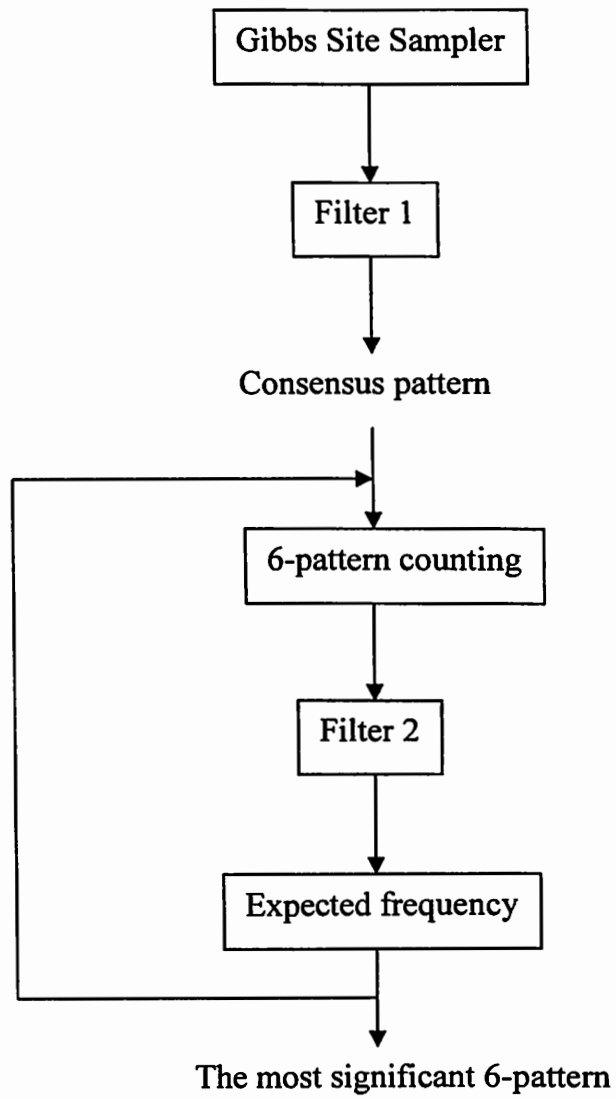


Figure 4.1 Flowchart of GS_PC.

Table 4.1 GS_PC as run on the Stormo CRP sequence set.

Cutoff	Fragmentation	Sites (false)	# Repeats	Expected frequency
0.10	9	22 (4)	1	4.8×10^{-21}
	12	22 (2)	2	5.0×10^{-23}
	15	22 (3)	7	1.2×10^{-27}
	18	20 (2)	3	1.0×10^{-27}
0.20	9	19 (3)	3	3.3×10^{-22}
	12	23 (3)	2	1.1×10^{-25}
	15	21 (0)	6	9.5×10^{-28}
	18	20 (2)	3	8.2×10^{-29}
0.30	9	17 (1)	2	1.2×10^{-23}
	12	22 (2)	1	1.6×10^{-27}
	15	21 (0)	6	9.5×10^{-28}
	18	20 (1)	1	8.2×10^{-29}
0.40	9	20 (4)	1	1.5×10^{-24}
	12	22 (2)	2	1.6×10^{-27}
	15	19 (0)	6	3.9×10^{-29}
	18	18 (0)	1	1.2×10^{-28}
0.50	9	16 (1)	1	1.9×10^{-23}
	12	20 (1)	1	1.9×10^{-26}
	15	19 (0)	7	3.9×10^{-29}
	18	19 (0)	2	2.4×10^{-28}
0.60	9	16 (2)	2	2.3×10^{-23}
	12	22 (2)	1	1.7×10^{-24}
	15	19 (0)	6	3.9×10^{-29}
	18	18 (2)	3	1.9×10^{-22}
0.70	9	15 (3)	1	5.1×10^{-22}
	12	15 (2)	1	3.9×10^{-23}
	15	17 (1)	4	1.9×10^{-25}
	18	17 (1)	1	3.4×10^{-26}

Table 4.2 The best result at each cutoff value from Table 4.1.

Cutoff	Fragmentation	Sites (false)	# Repeats	Expected frequency
0.10	15	22 (3)	7	1.2×10^{-27}
0.20	15	21 (0)	6	9.5×10^{-28}
0.30	15	21 (0)	6	9.5×10^{-28}
0.40	15	19 (0)	6	3.9×10^{-29}
0.50	15	19 (0)	7	3.9×10^{-29}
0.60	15	19 (0)	6	3.9×10^{-29}
0.70	15	17 (1)	4	1.9×10^{-25}

The whole process is repeated for each 6-pattern obtainable from the simplified consensus pattern. Finally, the 6-pattern with the lowest expected frequency is reported, together with the corresponding site alignments.

4.3 Results and discussions

The program implemented as described in Section 4.2 is run on the Stormo CRP sequence set, and the cutoff at both filtering points are varied between 0.10 ~ 0.70, the fragmentation in Gibbs Site Sampler is set at 9, 12, 15, and 18 columns, respectively. Each same parameter setting is run 10 times, and the most statistically significant, as measured by the least expected frequency, 6-pattern is reported. The results are recorded in Table 4.1. Table 4.2 summarizes the best result at each cutoff value.

It is obvious from Table 4.2 that the parameter settings with cutoff = 0.20 or 0.30, fragmentation = 15 columns give the best results, each finding 21 BSs and none being

false positive. More impressively, the same best results are repeated six times out of 10 runs. This indicates that the proposed combined algorithm is pretty steady. It also suggests a usage of the algorithm: run the algorithm, and report the result that repeats itself three times first.

The next best results are parameter settings with cutoff = 0.40 ~ 0.60, fragmentation = 15 columns (again). This setting gives 19 sites, none being false positive. Again, the results are pretty steady; each repeats itself at least six times out of 10 runs. It is not surprising that less sites are found with these parameter settings, since as the cutoff value is increased from 0.20 ~ 0.30 to 0.40 ~ 0.60, some true but weak BSs are filtered out. Also note that in all these five cases, fragmentation = 15 columns is repeatedly the best choice by fragmentation. This is in accordance with the biological model. As mentioned earlier, the binding sites in prokaryotes are typically 20 bp long (Stormo and Hartzell, 1989). Some of the positions are less conserved than others. At lower width value such as 9, some positions that are critical to the function of the binding sites and “information rich” (Neuwald *et al.*, 1995) are excluded from the process, thus leading to more random hits. As the width increases, the situation ameliorates, and the results improve. But at width 18, when some non-critical positions are included in the information content calculation process, this increases the chances that some sites that lack critical positions, yet still score above the cutoff value, to be included in the final output. As stated in Chapter 3, some positions are critical and must be reserved for a BS to act.

The Gibbs sampler is one of the most popular and accurate packages in identification of biosequence motifs. The authors maintain a website, <http://bayesweb.wadsworth.org/gibbs/gibbs.html>, where the latest Gibbs sampler (GS

new) is accessible. We submitted over 1,000 datasets to it, specifying different parameter settings in order to find the best performance of it, and the best parameter setting. The best performance and the best parameter setting for the Stormo CRP sequence dataset are shown in Table 4.3.

It is clear that our combined approach, GS_PC, considerably improves the performance of the old Gibbs sampler (GS old). In this specific dataset of Stormo CRP, GS_PC even outperforms the latest Gibbs sampler.

We then try our algorithm on two other datasets, LexA and purR, using the best parameter setting obtained from Stormo CRP. The results are shown in Table 4.4. In both datasets, the latest Gibbs sampler performs a little better than GS_PC.

So, overall, our combined approach, GS_PC, performs comparatively to the latest Gibbs sampler.

4.4 Conclusion and future work

We present a novel combined approach in the identification of TFBSs in prokaryotes that effectively reduces the candidate pattern space to be searched, and make feasible an enumerative scheme in this area. To our best knowledge, this is the first such combined approach.

GS_PC considerably improves the performance of the old GS, and is in the cases tested performing comparatively to the latest Gibbs sampler, one of the most accurate tools. Compared to alignment-based approaches, enumerative approaches are fast, and therefore GS_PC are especially valuable when handling large datasets.

Table 4.3 The best performance and best parameter setting of the old and new Gibbs sampler as run on the Stormo CRP dataset^a.

	Parameter settings	Total sites (false)
GS old	S_F_Len10 ^b	17 (1)
GS new	M_F_Len16_n10 ^c	19 (1)
GS_PC		21 (0)

- Over the following parameter settings: (i) Gibbs Site Sampler or Gibbs Motif Sampler, (ii) fragmentation, non-fragmentation, number of columns = 10, 12, ..., 30; or local search, number of columns (Gibbs new) = 10, 15, ..., 30, and (iii) expected number of sites (Gibbs Motif Sampler or local search) = 10, 20, 30. Each parameter setting was repeated six times.
- Gibbs Site Sampler, fragmentation = 10.
- Gibbs Motif Sampler, fragmentation = 16, expected number of sites = 10.

Table 4.4 The best performance and best parameter setting of the new Gibbs sampler as run on the LexA and purR datasets^a.

	GS_PC			GS new ^a
	Fragmentation	Cutoff	Sites (false)	Total sites (false)
LexA	15	0.30	18 (1)	19 (1) ^b
		0.50	18 (1)	
purR	15	0.30	17 (2)	15 (0) ^c
		0.50	18 (2)	

- Same as in Table 4.3.
- Gibbs Motif Sampler, fragmentation = 16, expected number of sites = 15.
- Gibbs Motif Sampler, fragmentation = 16, expected number of sites = 16.

Neither Gibbs sampler nor our approach is ideal. In particular, neither approach takes into consideration of information other than the primary sequences, such as the structural information of the DNA sequences. Chromatin-induced DNA TFBS binding is now widely considered an essential aspect in meditating the binding of DNA protein BSs (see Wasserman and Krivan, 2003 for a review). Incorporation of such information will bring the TFBS identification algorithms to a whole new level.

REFERENCES

1. Alder, H. L. and Roessler, E. B. (1972) Introduction to probability and statistics. W. H. Freeman and Company. Fifth ed.
2. Bailey, T. L. and Elkan, C. (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learn.*, 21, 51-80.
3. Benos, P. V., Bulyk, M. L. and Stormo, G. D. (2002) Additivity in protein-DNA interactions: how good an approximation is it. *Nucleic Acids Res.*, 30, 4442-4451.
4. Berg, O. G., and von Hippel, P. H. (1988) Selection of DNA-binding sites by regulatory proteins. *J. Mol. Biol.*, 200, 709-723
5. Brazma, A., Jonassen, I., Vilo, J. and Ukkonen, E. (1998) Predicting gene regulatory elements *in silico* on a genomic scale. *Genome Res.*, 8, 1202-1215.
6. Brazma, A., Jonassen, I., Eidhammer, I. and Gilbert, D. (1998) Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.*, 5, 279-305.
7. Bucher, P., Fickett, J. W. and Hatzigeorgiou, A. (1996) Computational analysis of transcriptional regulatory elements a field in flux. *Comput. Appl. Biosci.*, 12, 361-362.
8. de Crombrughe, B., Busby, S. and Buc, H. (1984) Cyclic-AMP receptor protein - role in transcription activation. *Science*, 224, 831-838.
9. DeRisi, J. L., Iyer, V. R. and Brown, P. O. (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278, 680-686.
10. Durbin, R., Eddy, S. R., Krogh, A. and Mitchison, G., *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge University Press (1998).
11. Duret, L. and Bucher, P. (1997) Searching for regulatory elements in human noncoding sequences. *Curr. Opin. Struct. Biol.*, 7, 399-406.
12. Fernandez de Henestrosa, A. R., Ogi, T., Aoyagim, S., Chafin, D., Hayes, J. J., Ohmori, H. and Woodgate, R. (2000) Identification of additional genes belonging to the LexA regulon in *Escherichia coli*. *Mol. Microbiol.*, 35, 1560-1572.

13. Fields, D. S., He, Y., Al-Uzri, A. Y. and Stormo, G. D. (1997) Quantitative specificity of the Mnt repressor. *J. Mol. Biol.*, 271, 178-194.
14. Frech, K., Herrmann, G. and Werner, T. (1993) Computer-assisted prediction classification and delimitation of protein binding sites in nucleic acids. *Nucleic Acids Res.*, 21, 1655-1664.
15. Ghosh, D. (1993) Status of the transcription factors database (TFD). *Nucleic Acids Res.*, 21, 3117-3118.
16. Grundy, W. N., Bailey, T. L. and Elkan, C. P. (1996) ParaMEME: a parallel implementation and a web interface for a protein motif discovery tool. *Comput. Appl. Biosci.*, 12, 303-310.
17. Hawley, D. K. and McClure, W. R., (1983) Compilation and analysis of *Escherichia coli* promoter DNA sequences. *Nucleic Acids Res.*, 11, 2237-2255.
18. Henikoff, S. and Henikoff, J. G. (1991) Automatic generation of protein blocks for database searching. *Nucleic Acids Res.*, 19, 6565-6572.
19. Hertz, G. Z. and Stormo, G. D. (1996) *Escherichia coli* promoter sequences. Analysis and prediction. *Meth. Enzymol.*, 273, 30-42.
20. Hertz, G. Z. and Stormo, G. D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15, 563-577.
21. Hertz, G. Z., Hartzell, G. W. and Stormo, G. D. (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.*, 6, 81-92.
22. Jonassen, I. (1997) Efficient discovery of conserved patterns using a pattern graph. *Comput. Appl. Biosci.*, 13, 509-522.
23. Jonassen, I., Collins, J. F. and Higgins, D. G. (1995) Finding flexible patterns in unaligned protein sequences. *Protein Sci.*, 4, 1587-1595.
24. Lashkari, D. A., DeRisi, J. L., McCusker, J. H., Namath, A. F., Gentile, C., Hwang, S. Y., Brown, P. O. and Davis, R. W. (1997) Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proc. Natl. Acad. Sci. USA*, 94, 13057-13062.
25. Lawrence, C. E. and Reilly, A. A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins Struct. Funct. Gen.*, 7, 41-51.

26. 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.
27. Lewis, L. K., Harlow, G. R., Grgg-Jolly, L. A. and Mount, D. W. (1994) Identification of high affinity binding sites for LexA which define new DNA damage-inducible genes in *Escherichia coli*. *J. Mol. Biol.*, 241, 507-523.
28. Maniatis, T., Ptashne, M., Backman, K. Kleid, D. Flashman, S., Jeffrey, A. and Maure, R (1975) Recognition sequences of repressor and polymerase in the operators of bacteriophage lambda. *Cell*, 5, 109-113.
29. McClure, W. R. (1985) Mechanism and control of transcription initiation in prokaryotes. *Annu. Rev. BioChem.*, 54, 171-204.
30. McCue, L. A., Thompson, W., Carmack, C. S. and Lawrence, C. E. (2002) Factors influencing the identification of transcription factor binding sites by cross-species comparison. *Genome Res.*, 12, 1523–1532.
31. Mehldau, G. and Myers, E. W. (1993) A system for pattern matching applications on biosequences. *Comput. Appl. Biosci.*, 9, 299–314.
32. Mulligan, M. E., Hawley, D. K., Entriken, R. and McClure, W. R. (1984) *Escherichia coli* promoter sequences predict in vitro RNA polymerase selectivity. *Nucleic Acids Res.*, 12, 789–800.
33. Neuwald, A. F. and Green, P. (1994) Detecting patterns in protein sequences. *J. Mol. Biol.*, 239, 698–712.
34. Neuwald, A. F., Liu, J. and Lawrence, C. E. (1995) Gibbs Motif Sampling: detection of bacterial outer membrane protein repeats. *Protein Sci.*, 4, 1618–1632.
35. Pearson, W. R. and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85, 2444-2448.
36. Pribnow, D. (1975) Nucleotide sequence of an RNA polymerase binding site at an early T7 promoter. *Proc. Natl. Acad. Sci. USA*, 72, 784-788.
37. Queen, C., Wegman, M. N. and Korn, L. J. (1982) Improvements to a program for DNA analysis: a procedure to find homologies among many sequences. *Nucleic Acids Res.*, 10, 449-456.
38. Robison, K., McGuire, A. M. and Church, G. M. (1998) A comprehensive library of DNA-binding site matrices for 55 proteins applied to the complete *Escherichia coli* K-12 genome. *J. Mol. Biol.*, 284, 241-254.

39. Rombauts, S., Florquin, K., Lescot, M., Marchal, K., Rouze, P. and Van de Peer, Y. (2003) Computational approaches to identify promoters and *cis*-regulatory elements in plant genomes. *Plant Physiol.*, 132, 1162-1176.
40. Sagot, M. F. and Viari, A., Double combinatorial approach to discovering patterns in biological sequences. In: D Hirschberg and G Myers, eds, *Combinatorial Pattern Matching, Lecture Notes in Computer Science (volume 1075)* (1996) pp. 186–208. Springer-Verlag
41. Schneider, T. D., Stormo, G. D., Gold, L., Ehrenfeucht, A. (1986) Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188, 415-431
42. Sinha, S. and Tompa, M. (2002) Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res.*, 30, 5549-5560.
43. Smith, H. O., Annau, T. M. and Chandrasegaran, S. (1990) Finding sequence motifs in groups of functionally related proteins. *Proc. Natl. Acad. Sci. USA*, 87, 826-830.
44. Smith, R. F. and Smith, T. F. (1990) Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Natl. Acad. Sci. USA*, 87, 118-122.
45. Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D. and Futcher, B. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell.*, 9, 3273-3297.
46. Staden, R. (1984) Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.*, 12, 505–519.
47. Staden, R. (1989) Methods for discovering novel motifs in nucleic acid sequences. *Comput. Appl. Biosci.*, 5, 293–298.
48. Stormo, G. D. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, 16, 16-23.
49. Stormo, G. D. and Hartzell, G. W. III (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA*, 86, 1183–1187.
50. Stormo, G. D., Schneider, T. D. and Gold, L. (1986) Quantitative analysis of the relationship between nucleotide sequence and functional activity. *Nucleic Acids Res.*, 14, 6661-6679.
51. Stormo, G. D., Schneider, T. D., Gold, L. and Ehrenfeucht, A. (1982) Use of the 'Perception' algorithm to distinguish transcriptional initiation sites in *E. coli*. *Nucleic Acids Res.*, 10, 2997-3012.

52. Suyama, M., Nishioka, T. and Oda, J. (1995) Searching for common sequence patterns among distantly related proteins. *Protein Engng.*, 8, 1075-1080.
53. Tatusov, R. L., Altschul, S. F. and Koonin, E. V. (1994) Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proc. Natl. Acad. Sci. USA*, 91, 12091-12095.
54. Thijs, G., Lescot, M., Marchal, K., Rombauts, S., Moor, B. D., Rouze, P. and Moreau, Y. (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics*, 17, 1113-1122.
55. Tompa, M. (1999) An exact method for finding short motifs in sequences, with applications to the ribosome binding sites problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Heidelberg, Germany, pp. 262-271.
56. van Helden, J., André, B. and Collado-Vides, J. (1998) Extracting regulatory sites from the upstream region of yeast by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.*, 281, 827-842.
57. van Helden, J., Rios, A. and Collado-Vides, J. (2000) Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res.*, 28, 1808-1818.
58. Vanet, A., Marsan, L. and Sagot, M. F. (1999) Promoter sequences and algorithmical methods for identifying them. *Res. Microbiol.*, 150, 779-799.
59. Wasserman, W. W. and Krivan, W. (2003) In silico identification of metazoan transcriptional regulatory regions. *Naturwissenschaften*, 90, 156-166.
60. Waterman, M. S., Arratia, R. and Galas, D. J. (1984) Pattern recognition in several sequences consensus and alignment. *Bull. Math. Biol.*, 46, 515-527.
61. Wingender, E., Dietze, P., Karas, H. and Knüppel, R. (1996) TRANSFAC: a database on transcription factors and their DNA binding sites. *Nucleic Acids Res.*, 24, 238-241.
62. Wishart, D. S., Boyko, R. F., Willard, L., Richards, F. M. and Sykes, B. (1994) SQSEE: a comprehensive program suite for protein sequence analysis. *Comput. Appl. Biosci.*, 10, 121-132.
63. Wolfertstetter, F., Frech, K., Herrmann, G. and Werner, T. (1996) Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithms. *Comput. Appl. Biosci.*, 12, 71-80.

APPENDIX A

THE THREE *ESCHERICHIA COLI* BINDING SITE

SEQUENCE DATASETS

This appendix lists the three *Escherichia coli* sequence datasets that are used in this study.

A.1 CRP sequence data

```
>1. aldA 1486060..1486255
aaattgcccg tttgtgaacc acttgtttgc aaacgggcat gactcctgac ttttatttct
gccttttatt ccttttacac ttgtttttat gaagcccttc acagaattgt cttttcacga
ttccgtctct ctgatgattg atgttaatta acaatgtatt caccgaaaac aaacatataa
atcacaggag tcgccc
```

```
>2. ansB(R) 3098748..3098922
tcctctatth taagacggca taactctttt ttatgccggt taattcttcg ttttggtacc
tgcctctaac tttgtagatc tccaaaatat attcacgttg taaattgtht aacgtcaaat
ttcccataca gagctaaggg ataatgcgta gcgttcacgt aactggagga atgaa
```

```
>3. araB 70049..70386 NOTE: -GenBank.
cgtttcactc catccaaaaa aacgggtatg gagaaacagt agagagttgc gataaaaagc
gtcaggtagg atccgctaath cttatggata aaaatgctat ggcatagcaa agtgtgacgc
cgtgcaaata atcaatgtgg acttttctgc cgtgattata gacacttttg ttacgcgtht
ttgtcatggc tttgggtccg ctttgthtaca gaatgcttht aataagcggg gttaccggtt
gggthtagcga gaagagccag taaaagacgc agtgacggca atgtctgatg caatathggac
aathggthtc thctctgaath ggtgggagta tgaaaagt
```

```
>4. araE 2980205..2980518 NOTE: -GenBank.
tttttctgc cagcagagag taagacathg tgaaaaaata cgtgaacaac tcacgcaggt
gtcaggtcgg aacacagcata aathatggath aaathgctgc gacathgctg thtgthgatgg
aththccaath thtcaaatha agthgaatha thgagathat thathaaccah cthathththac
agcagathaaa athcataaag thcathath gathathath atggathath thcataacath
```

gatatggatt atgatgatct acaggtataa aaaaccctgc catgcggcag ggtcataaaa
gtaagaagaa tgaa

>5. araF(R) 1984152..1984947

acctaatac cttatatcca gaagtggaga ggtgcaagat aatcaaaac accataatta
ctgtttgcag aataattcct ttctgccggt ccgaaatgtg atcgtggagt caattctgac
gatgatctga aataagaata acgtctgaac gggataaga aatgcttaa atcatcaagt
ttatatattt ttacatttca ttgatgaaat caatgtaact gcaatgaatt ttaatgatag
tgctttttgt atcttgttga tattaatcaa tgaatttgaa tcatgatga agtgatgatt
ttataaaacg ttttttcatt tttgcgagat ggctctcatt atacgtgttc tgtaattaa
tgagcacagt gataattaat gacgaaatga atgacgtgca tttccacat ctttgagttg
cggttattac accatttcaa aaactcaacg ccaggtaatg cggcctattg actggttaaa
agaagacat cccgcatggg taccaaagac aacaaggatt tccaggctaa tcttatggat
taatctgctg tgcattcgac aatttgtctg acaaatggc tttcccttat gtcttttccc
gctaaattta tgcacgttct cactgtaatt ctgcatgtg atattgctct cctatggaga
attaatttct cgctaaaact atgtcaacac agtcacttat cttttagtta aaaggtaatg
ctttgttttc cgattaattt aacgaatgtc attcgttttt gcctacaca aaacgacact
aaagctggag agaacc

>6. araJ(R) 411706..411830

ctattcagca ggataatgaa tacagagggg cgaattatct cttggccttg ctggtcgtta
tcttgcaagc tatcacttta ttggctacgg tgattggtag ccgttctggt ggttgatgag
gtggt

>7. cdd 2229735..2229863

aatttgcgat gcgtcgcgca tttttgatgt atgtttcacg cgttgcataa ttaatgagat
tcagatcaca tataaagcca caacggggtc gtaaactggt atcccattac atgattatga
ggcaacgcc

>8. crp(R) 3483456..3483756 NOTE: -GenBank.

gcgcgggtat cctctgttat aagctttctc cagagccaga taacgccgct gtctctggat
tgccgaaata tgcttcccgc tacctgggaa ggggctatca actgtactgc acggtaatgt
gacgtccttt gcatacatgc agtacatcaa tgtattactg tagcatcctg actgttttag
catagctttc gctttgtgtc tcttggtgtc tcgcttcagc atgaccagg tcgccttccg
ttgcgcgatt tggtagtac gcgtactctg tcaggaaaat tgacgcagtg gagtagcaaa
a

>9. cyaA 3988401..3988765

ccgtgggtcca tcctaacatc cttgccagag tgatgtcagt gttgtggtga aacgtagacg
cctgcgcaaa ccgtaaaatg aggtctggca gtggatcctg acaggcgttt cacgccggtg
taataaggaa ttacagaga ataaacgggt ctacacttgt atgtagcgca tctttcttta
cggatcaatca gcaagggtgt aaattgatca cgtttttagac cattttttcg tcgtgaaact
aaaaaaacca ggcgcgaaaa gtggtaacgg ttacctttga catacgaaat atcccgaatg
ccgcgtgtta ccgttgatgt tggcggaatc acagtcatga cgggtagcaa atcaggcgat
acgtc

>10. cytR 4122037..4122191 NOTE: -GenBank.
actcactcct cgcctggcac gtcaggcgta ctacatccat gtttacttca catcggcaac
atTTTTtagca gatagcgcgt gaaaacgggt acagaatttt catgaaaagt gtgatgaata
ttgaattttt cgatccgcct cgcacgtgga gcggt

>11. dadA 1236465..1236793
acgcgcacct cattgttgtc ggcgctctct gtgtggagca ctcatttca agcatagaac
acctgttaaa aaccgcgtcg ccggagaatt ttttctttg cgatttctta ttatcagagt
gccactaatc cgcttctgaa cggaaattta tgctggataa aaagggcggt cagcaggaga
tactaaagac gccatattgc cgcagagtca gggagatgtg agccagctca ccataaaaaa
gccgcatggt gaataatatt ttcaactgag ttatcaagat gtgattagat tattattctt
ttactgtatc taccgttatc ggagtggct

>12. deoC 4614635..4614891
cttcttttcc ttttatgccg aaggatgagc gccattgtaa gaagtttctg gatgttctact
ttgatcctga tgcgtttgcc accactgacg cattcatttg aaagtgaatt atttgaacca
gatcgcatta cagtgatgca aacttgtaag tagatttctt taattgtgat gtgatcga
gtgtgttgcg gagtagatgt tagaatacta acaaactcgc aaggatgaatt ttattggcga
caagccagga gaatgaa

>13. flhD(R) 1976231..1976541
gttggagtca ttaccattt atgttaagta attgagtgtt ttgtgtgatc tgcacacgc
attattgaaa atcgcagccc cctccggtg tatgtgcgtg tagtgacgag tacagttgag
tcgatttagg aaaaatctta gataagtga aagaccatt tctatttgta aggacatatt
aaaccaaaaa ggtggttctg cttattgcag cttatcgcaa ctattctaata gctaattatt
ttttaccggg gcttcccggc gacatcacgg ggtgcggtga aaccgcataa aaataaagt
ggttattctg g

>14. fur(R) 709870..710157
tgtgatgcgg cgtagactca tgtctacgcc gtattaatag ataatgcaa tcaaaataat
tgctacaaat ttgtaacttt tgctgttgta cctgtacaat gtcccgggtg tcaagtggcc
ttgccgttgt aaatgtaagc tgtgccacgt ttttattaac aatatttgcc agggacttgt
ggttttctatt taggcgtggc aattctataa tgatagcat tatctcaaga gcaaatctg
tcacttcttc taatgaagtg aaccgcttag taacaggaca gattccgc

>15. galE 791279..791538 NOTE: -GenBank.
aattcgctcc attaggctta tggatgaaa taaccatagc ataacaaaga tgcgaaaagt
gtgacatgga ataaattagt ggaatcgttt acacaagaat ttagccggtt tttatgcgag
attaagtgat tataaaacag agggtttatg aatgattgag ctttttatct gaaaaagac
gcggtttcat gcctgcatgc gtcgaaccgt tggccggaga gggtgctaag gccgcctccg
gcaaggtcag cactaccgac

>16. gcvT(R) 3048688..3049134
aaatttctcc tctgttgttt atttgatacc catcacactt tcactctccg gtttttctgc
cgggagattt tcctcatttg aaataaacta atttcacctc cgttttcgca ttatattttc
taatgccatt atttttgat ttagtgtttt ttgacatttt ttagctctt aatattgtct

tattcaaatt gactttctca tcacatcadc tttgtataga aactggtgta ttttttggtt
ttttattctg tcgcgatttt tgcatttttt aaccataagc taatgtgatg atcaatttta
ccttatgggt aacagtctgt ttcggtggtg agttcaggca aaagagaacg attgcggttg
ggaccgggag tggctccgat gctgggtttc gtggtgataa tttcaccatg aaaaagttgt
cagccccgct tattcaatga ggacaag

>17. glpA(R) 2350395..2350668 NOTE: -GenBank.
tgttatccct ctgaagttcg ttttttacca tttagccata gtaaaaacat gaattgtttg
atttcgcgca tattcgctca taattcgaaa gtgaaacgtg atttcatgcg tcattttgaa
cattttgtaa atcttattta ataatgtgtg cggcaattca catttaattt atgaatgttt
tcttaacatc gcggcaactc aagaaacggc aggttctctc actgaatcag gctgttaatc
ataaataaga ccacgggcca cggaggctat ca

>18. glpD 3559457..3559647
gcgtctctct ttctttacaa acaagtgggc aaatttaccg cacagtttac gtcgaagcgg
cagataaacg ccataatggt atacatatca ctctaaaatg ttttttcaat gttacctaaa
gcgcgattct ttgctaatat gttcgataac gaacatttat gagctttaac gaaagtgaat
gagggcagca t

>19. glpF(R) 4115671..4116094
acctctcctg aattgcaagg cgttgatgga taaaaatcct cgtccccgatt accggtgacg
ccttaataaaa tacgagcgca ctttagttag ctccgattgt atgaagccgc gccatcgctg
tccagcggca cgccttgagc attacggttt gccacacttt tcatccttct cctggtgaca
taatccacat caatcgaaaa tgtaataaaa tttggtgagc gaatgatcta acaaacatgc
atcatgtaca atcagatgga ataaatggcg cgataacgct cattttatga cgaggcacac
acattttaag ttcgatattt ctcgtttttg ctcgttaacg ataagtttac agcatgccta
caagcatcgt ggaggtccgt gactttcacg catacaacaa acattaactc ttcaggatcc
gatt

>20. guaB(R) 2632091..2632251
gtgagcgaga tcaaattcta aatcagcagg ttattcagtc gatagtaacc cgcccttcgg
ggatagcaag cattttttgc aaaaaggggt agatgcaatc ggttacgctc tgtataatgc
cgcggaacaa tttattaacc actctggtcg agatattgcc c

>21. ilvB(R) 3850412..3850516
ggactggaac aacacacgat tccaaaaccc cgccggcgca aaccggggcg ggtttttcgt
ttaagcacct cccggaaagt cggcccagaa gaaaaggact ggagc

>22. lacZ(R) 365530..365651
gcgcaacgca attaagtga gttagctcac tcattaggca cccagggtt tacactttat
gcttccggct cgtatgttgt gtggaattgt gagcggataa caatttcaca caggaaacag
ct

>23. malE 4243999..4244362 NOTE: -GenBank.
aatctatggg ccttgttggg gaagtgtctg tgaaaacacc taaacggact ctagtcttctt
tatacggcaa cctctttcca tctccttgc cctacgccc caccgtcgtc ttgtgtgatc

tctgttacag aattggcggg aatgtggaga tgcgcacata aaatcgccac gatttttgca
agcaacatca cgaaattcct tacatgacct cggtttagtt cacagaagcc gtgttctcat
cctccccgct cctcccccat aaaaaagcca ggggggtggag gatttaagcc atctcctgat
gacgcatagt cagcccatca tgaatgttgc tgtcgatgac aggttggttac aaagggagaa
gggc

>24. malS 3734807..3735125

caggtctcct ggctcggattt aatcattcca acaccttata tttttcacia atttgagagt
tgaatctcaa atcatatcaa aaatagctgt caagagcacc ccaaggaata gtccaaatct
gaaactatgt cacgtgttaa cgattcagat tggcgctaaa tcgcagaaaa tgtggggggt
atcgcaaaat tcagccgttt tttgcgcgag atcgctcacc cttgcttctc atcctgtgga
cttaccgctc agggatgagt tttgtttggc ttatcgctgg caaactgtct gaaatcgcag
caataaggac tcatccgcc

>25. malT 3550107..3550717

aggagtcca cttttcttag attttcaaca caacgttatc gctagtttgc caggctcgat
gttgacctc ctcatcctgc gggggattag gcagggagga gttgcgggga tgagcaagga
aatgtgatct caaccactta aagctagtgc aaaccacagg attagcatca aatcaatgca
atacagcgca gaaaatctgt atctaagtgc aaaaaatggc cgttgcgtat tttcaaaaag
cggaaggtaa ctctataaat taagtaaagg agtgaaacag tttcataagt aaaatatcca
gtgtgctcca tctcattctt aatagattta ttaagatcat ctttttagat ggcactttca
tcaggaatga agaagaaacc cttgcttaaa tgaatctgat gaacataagg gaaaccagta
ttcacgctgg atcagcgtcg ttttaggtga gttgttaata aagatttggga attgtgacac
agtgcaaatt cagacacata aaaaaacgtc atcgcttgca ttagaaagg tttctggccga
ccttataacc attaattacg aagcgcaaaa aaaataatat ttcctcattt tccacagtga
agtgattaac t

>26. manX(R) 1899610..1900071 NOTE: -GenBank.

ttgctacctc ctttattatc gttaacacct caacgtgcca gatgtatttt tgaatcgctc
tccacaatcg aatcgattca gataagggt acaaccaaaa acccctgcgt ttcgcgattt
atthtagata tcgaaaaaat tattttatgt gatgaagatc cgtaatttaa ctttcgatta
gcagaaattt cgaaaggtaa aatatccttg tcacattcgt ttgcaaagga aggtaaatct
ttgccaatc agaggcgtct ctgatatgtt taactcccgt ttaacaacca tggagtatag
ggcagtagcc cgcagtatgg atcgtcaccg acgtcatttc agcatcaggc cttttaacgc
ctgcctttct ggcactctat gccgcacctt tcgtttgcatt tttgtcgta cgcctgcatt
atthctggcg tcgaatagct attccttaag caggagcttg tc

>27. melA 4339207..4339490

ggatggctct ctttcttggga atatcagaat tatggcagga gtgagggagg atgactgcga
gtgggagcac ggttttcacc ctcttcccag aggggagcagg ggactctccg agtatcatga
ggccgaaaac tctgcttttc aggtaattha tcccataaa ctcagattta ctgctgcttc
acgcaggatc tgagtttatg ggaatgctca acctggaagc cggaggtttt ctgcagattc
gcctgccatg atgaagttat tcaagcaagc caggagatct gcat

>28. mtlA 3769372..3769907

atctgcctcg gattcacgtt tatcagtggt gtttttgggc tggcagccag aagggagtca

ggctgatatt ttgacaataa tccgggttcg cgattctcgc cataacacca aagaataatt
tttagagggtg atgagttgct tagttacata acgattgtat gacgaaggca taacatgctg
tagatcacat caggtgaacg ccgtaagaaa atatcttggtg attcagatca caaagattca
acaaccatc aaaacaaaaa tgtgacacta ctcacattta aatgccattt ttagcgaaaa
tcgccgcctt gttgcttttt tacacaagcg ttttgtgatg aacgtcacgt caattacctc
tctaccccct atatttatgt gattgatatc acacaaaagg ccgtcgactg gacagttaac
cgattcagtg ccagatttcg cagtatctac aagggtccggc tacctctgcc gccacattaa
caaaaaacct cgggcttcca gcctgcgcga cagcaaocat aagaaggggt gttttt

>29. nagB(R) 702835..703166

cttattcccc ctacgagaac cctatttggc tcgtttcaag ccgtatTTTT attttgctgc
aaattgtact gccgatgttc tgtaatcaga ttgtagatc atctgctaca gagtgtgtga
aaatttaatt cgtatcgcaa attaaacgcg tgtcttttgt gagttttgtc accaaatatac
gttattatca ctccctttta ctggctaaac cagaaaactt attttatcat tcaaaaaatc
aggtcggatt gacgcctgtc tgcgcaaatc caggttacgc ttaaagatgc ctaatccgcc
aacggcttac attttactta ttgaggtgaa ta

>30. nupG 3103531..3103683

ttccattaac cgcccctgac gatgctcagg ggcaaaaatg ttatccacat cacaatttcg
ttttgcaaat tgggaatggt tgcaattatt tgccacaggt acaaaaaaac cagtccgcga
agttgataga atcccatcat ctgcacaggt caa

>31. ompA 1019277..1019632 NOTE: -BenBank & -DPInteract.

tttttgccgc tcgttatcat ccaaaatagc ccatgaatat ctccaacgag ataacacggg
taaactcttc accgggggat ctgctcaata ttaactctac cgatatcttc ggcttatgcc
gagcaccctt ggcgatgtaa agtctacaac gtagttgaaa acttacaagt gtgaactccg
tcaggcatat gaaaaaaaaag tcttgataa ggtatgttta atcttttttg tcagcgacaa
ttacagaag agaatcgcgg aaaccgcttc agacaagcct ccgcaaggaa aattagtcac
gactgaaagc attggctggg cgacaaaaaa agttccagga ttaatcctaa atttac

>32. ompR(R) 3534223..3534413

attgggtata acgtgatcat atcaacagaa tcaataatgt ttcgccgaat aaattgtata
cttaagctgc tgtttaatat gctttgtaac aatttaggct gaaattcata ccagatttag
ctgggtgacga acgtgagctt ttttaagaat acacgcttac aaattgttgc gaaccttgg
gagtacaaac a

>33. pckA 3530078..3530457

gttgggtatc cagaatcaaa aggtgggtta attatcgcac ccgggcagta gtatTTTgct
tttttcagaa aataatcaaa aaaagttagc gtgggtgaatc gatactttac cggttgaatt
tgcacatcaatt tcattcagga atgcgattcc actcacaata ttcccgcac ataaaccaag
atttaacctt ttgagaacat tttccacacc taaaatgcta tttctgcgat aatagcaacc
gtttcgtgac aggaatcacg gagttttttg tcaaatatga atttctccag atacgtaaat
ctatgagcct tgctgcgggtt aacaccccc aaaaagacttt actattcagg caatacatat
tggttaagga gcagtgaat

>34. ppiA(R) 3489935..3490204

gattggcctg cgttcaaaaa taaaatggca tagcgggata tgccgogagc gggcgatttt
agggtatttt gtgatctgtt taaatgtttt attgcaatcg gttgctaaat tgcattttaa
gaggtgattt tgatcacgga ataaaaagtg atcgtcaggt tacatatatt tcagatacgt
aaaattaggt aaagggatgg ccttgttctt gaaggctatt tagaatctct tcacttgctt
tttttctgct ctgtttgtta aggaaatctc

>35. proP 4327817..4328079
ataatcagtt acatcaatga gtcctaaacg aaatccatgt gtgaagttga tcacaaattt
aaacactggg agggtaaaaa ggtcattaac tgcccaattc aggcgtcaac tggtttgatt
gtacattcct taaccggagg gtgtaagcaa acccgctacg cttgttacag agattgcatc
ctgcaattcc cgctcccctt ttgcgggcgt cgcgctgatt tttctggcgt ttgcggaat
gggccaactc tgcgaggaaa gct

>36. ptsH 2531401..2531783
tgccagcttg ttaaaaatgc gtaaaaaagc accttttttag gtgctttttt gtggcctgct
tcaaactttc gccctcctg gcattgattc agcctgtcgg aactgggtatt taaccagact
aattatthttg atgcbgaaa ttaatcgtta caggaaaagc caaagctgaa tcgattttat
gatttggttc aattcttctt ttagcggcat aatgtttaat gacgtacgaa acgtcagcgg
tcaacaccog ccagcaatgg actgtattgc gctcttcgtg cgtcgcgtct gtaaaaaact
ggcgctaaca atacaggcta aagtcgaacc gccaggctag actttagttc cacaacacta
aacctataag ttggggaat aca

>37. rhaS 4095030..4095316
aatgtgatcc tgctgaattt cattacgacc agtctaaaaa ggcctgaat tcgogacctt
ctcgttactg acaggaaaat gggccattgg caaccaggga aagatgaacg tgatgatgtt
cacaatttgc tgaattgtgg tgatgtgatg ctcaccgcat ttctgaaaaa ttcacgctgt
atcttgaaaa atcgacgttt tttacgtggg tttccgtcga aaatttaagg taagaacctg
acctcgtgat tactatthtc ccggtgtgac gacatcagga ggccagt

>38. rhaT(R) 4098107..4098390
attcatctcc agtattgtcg ggcggccgat tgtaatgcc gcgtaagcag ttggttcatt
atagtttaatt aatgatatt gaaaatgatt atcaatgccg tacttttctg aagggtatgg
ttttgcagga aatgcccga gatgtgaagc aatcaccca cttaatgccg tgattgccag
taaactgaca acggcggcaa caggcgaag gttaatcgac agcacgattt ttacactcat
ctcgtcggag atgtgacgcg acgaaaaatg atgaggataa gaag

>39. rpoH(R) 3598415..3598658
taaaagcgtg ttatactctt tccctgcaat gggttccgta gcagggaaaag agaccccgtt
gtctcttccc ggtatttcat ctctatgtca cttttgtgc gtaatttatt cacaagcttg
cattgaactt gtggataaaa tcacggtctg ataaaacagt gaatgataac ctcggtgctc
ttaagctctg gcacagttgt tgctaccact gaagcggcag aagatcga ttgagaggat
ttga

>40. srlA(R) 2823600..2823854 NOTE: -GenBank.
tgttctctcc ttcaggattt attgttttat taccaaacgg caacctaatc taatcagatt
gaaagattta aaagtgttat tttgatcgca aatgaaaga taaatatttt aatttgaag

tttgaataaa aggatagcga ggggaatgag ttgagttatg taaagtccgt atcggggcagt
gactaccgct tccttgtgcg gggcgatgat ctttaccata cttgcccctg gttgaatctg
ttaaatggac ccctc

>41. tdcA(R) 3264707..3264894

tttttttgac aaaaatcagg gtttatgctg atttttatac ttttaacttgt tgatatttaa
aggtatttaa ttgtaataac gatactctgg aaagtattga aagttaattt gtgagtggtc
gcacatatcc tgttcatttc attttgatac acttcatgcc gtcaatgagg taattaacgt
aggtcggt

>42. tna 3886139..3886343

tttgcccctc tgtagccatc accagagcca aaccgattag attcaatgtg atctatttgt
ttgctatatc ttaattttgc cttttgcaaa ggcatctct cgtttattta cttgtttttag
taaagtatgg tgcttgcata tatatctggc gaattaatcg gtatagcaga tgtaatatc
acagggatca ctgtaattaa aataa

>43. tsx(R) 431238..431535

acgtatttctg ggacgatttt gtgcgtcccg caacatcttt ccccgtcatt ttgttactct
gcttacatca cctggattga tagtaaaagt ttgcaacaag ggcgaaagtc agtacaatcc
ccgcccgaat gtgtgtaaac gtgaacgcaa tcgattacgt aatgataga actgtgaaac
gaaacatatt tttgtgagca atgattttta taataggctc ctctgtatac gaaatattta
gaaacgcaat ttgcgccttt ttcactcccg caagggattt tcaaacagtg gcatacat

>44. udp 4013720..4014017

gcagcaggtg caaatccaga ttgttgtggt gttgccatgg tattctccgt acctataaaa
aatggtgctc aatgttaact atagtcagca tgcaacaaat cacattgcct gaatcggctc
atcttttatg cagtcctgca gaatgaaggg tgatttatgt gatttgcac acttttgggtg
ggtaaattta tgcaacgcat ttgcgtcatg gtgatgagta tcacgaaaaa atgttaaacc
cttcggtaaa gtgtcttttt gcttcttctg actaaaccga ttcacagagg agttgtat

>45. uxuA 4548866..4549204

aacgttttac cttacctggt tgaaccggtg ttattttggg cgatatgtta tgtaaattgg
tcaaccattg ttgcatgaa tgcacatcc tctgatcaat aaccatcgat taccctttgc
tgcaatttgc agcaacaacc atgagagtga aattcttgtg atgtggttaa ccaatttcag
aattcggggt gacatgtctt accaaaagggt agaacttata cgccatctca tccgatgcaa
cgccacggct gcggtctggt tgttcacccg gatacctaaa caactccagg gttccgctc
tctttgctgt ggaaccact atgtgaaaga ggaaaaatc

A.2 LexA sequence data

>1. b1741 1821310..1821538

taatttgcac atattggatt gtgcgaaaaa gagtaatttg ttcacgccgg atgcggcgtg
aacgccttat tcgacctata aaactatgca aattcaatac attgcaggag tcgaataggc

ctgacaggcg tagcacgtca gacgggtgtaa cctttgtcat cgacccgctt cttttttaat
cgcttcccgc ctgttacact ggatagataa ccagcattcg gagtcaaca

>2. dinG 832174..832292

taaaccgcat tatgttggtg gttattgca gccgctttcc agaaacagaa aaaccattac
ccctgaaaac cgaaaaatgc cacaatattg gctgtttata cagtatttca ggttttctc

>3. dinI(R) 1120711..1120783

aaatcgtagc ttcctgttgt cattaggtta ttttacctgt ataaataacc agtatattca
acagggggct att

>4. ftsK/dinH 932313..932446

cacggaacag gtgcaaaatc ggcgtatttt gattacactc ctgttaatcc atacagcaac
agtactgggg taacctggta ctgttgctcg ttttagcatc gggcaggaaa agcctgtaac
ctggagagcc tttc

>5. lexA 4254585..4254693

cccttccaga attcgataaa tctctggttt attgtgcagt ttatggttcc aaaatcgctt
tttgctgtat atactcacag cataactgta tatacaccca gggggcgga

>6. polB/dinA(R) 65781..65854

tgactgtata aaaccacagc caatcaaagc aaaccaggct atactcaagc ctggtttttt
gatggatttt cagc

>7. recA(R) 2821793..2821871

tactgtatga gcatacagta taattgcttc aacagaacat attgactatc cggattacc
cggcatgaca ggagtaaaa

>8. recN 2749731..2749815

ttttacgcca gcctctttac tgtatataaa accagtttat actgtacaca ataacagtaa
tggtttttca tacaggaaaa cgact

>9. ruvAB(R) 1944001..1944175

tgaatatgta atattaaaat atttgcttcc aatataacct gtagaataaa ttatactgtg
ccatttttca gttcatcgag acacctcgca agttttcttc atccttcgct ggatatctat
ccagcatttt tttatcatac agcattatct ttgattcatt acgcaggagc gtcatt

>10. sulA(R) 1020143..1020360

aaaattcctt ttaaaatcat aacataaaaag aatgattcac attaacggat ccgtaacta
cgaaaatagg caacttattc ttaaggggca agattaattt atgttttccc gtcaccaacg
acaaaatttg cgaggctctt tccgaaaata gggttgatct ttgttgctac tggatgtact
gtacatccat acagtaactc acaggggctg gattgatt

>11. umuD 1229624..1229989

tcgcctcttt aatatataa attgtaatga aactcctggt ttacaactat taataaattt
tacttcatct aattcatagt tagccgggcg ggatgcgtca atgtctttat ttctattaat

atgataaata tcaaacaatg tttaatgtca ttatggcgaa tgcttctatt ctatTTTTTTA
gccgggtgat atTTTTtCatt tctgctggat gagcgtcgtc gccagaaggc cacgtgagca
caagataaga gaacgaaaaa tcagcagcct atgcagcgac aaatattgat agcctgaatc
agtattgatc tgctggcaag aacagactac tgtatataaa aacagtataa cttcaggcag
attatt

>12. uvrA(R) 4271451..4271703

gttcgtgtct cctgaaaaaa atcgttctga ataagtgtaa acgcgcgatt gtaccattac
caatagcgct tttactatgt tgtgacctcg gttccgggaa acaaacctgg ccagacattg
ttacacaaca ctccgggtaa tgcattccaa tactgtatat tcattcaggc caatttTgtgT
cataattaac cgtttTgtgat cgccggtagc accatgccac cgggcaaaaa agcgttTaat
ccgggaaagg tga

>13. uvrB 812171..812748

ccattctgta tttggTtaaa ttgcgagcga gatcgcgtct tcgattgact gcaattTtaac
caattaaatt ctaaaataat cacgaaaaaa attttacttc cgctcatgc ggCGaatgtg
ggaattgccc aggcggcggg ggataggggc tggagacagt tatccactat tcctgtggat
aaccatgtgt attagagtta gaaaacacga ggcaagcgag agaatacgcg gcttgCacgc
gaattggcgt taaagacggc tcaaagaaat atcttttatt ttttaactgg ttagataaat
gcaatggcag tCactgaaca ggcattctct gccataaaac tgcattcact catcttgaca
aatgtTtaaaa aagccgttgc tttggggata acccggtaaG gccggagttt tatctcgcca
cagagTaaat tttgctcatg attgacagcg gagtttacgc tgtatcagaa atattatggT
gatgaactgt ttttttatcc agtataattt gttggcataa ttaagtacga cgagTaaat
tacatacctg cccgccaac tccttcaggc agcgactc

>14. uvrD 3995513..3995595

tcagcaaate tgtatatata ccagctttt tggcggaggg cgttgcgctt ctccgccccaa
cctatTTTTTA cgcggcgggtg cca

>15. yjiW(R) 4577468..4577637

acattatTTT ctggcgCacc tttccgggtgc gctttttatt atttcacgcc aatcataacc
cacataaata tattTaaatc attccagaaa ttgccattt tattctattt ttagctggac
tttccccata tttactgatg atatatacag gtatttagcg cggTgcggat

A.3 purR sequence data

>1. carA 29196..29650

ccacaaaata tttgttatgg tgcaaaaata acacatttaa tttattgatt ataaagggct
ttaatTTTTg gcccttttat ttttggTgtt atgtttTtaa attgtctata agtgccaaaa
attacatgtt ttgtctctcg tttttgtTgt tttaatgtaa attttgacca tttggTccac
ttttttctgc tcgTTTTtat ttcatgcaat cttcttgctg cgcaagcgtt ttccagaaca
ggTtagatga tctttttgTc gcttaatgcc tgtaaaacat gcatgagcca caaaataata
taaaaaatcc cgccattaag ttgactTTta gcgccccatc ctccagaatg ccgcccgtttg

ccagaaattc gtcggtaagc agatttgcac tgatttacgt catcattgtg aattaatatg
caaataaagt gagtgaatat tctctggagg gtgtt

>2. codB 353817..354145

gttattgtcg gatgcgtcgc gcggtgcac cggcactgtg tgccgatgcc tgatgcgacg
ctgacgcgtt ttatcatgcc tacggacctg aaccgtaggt cggataaggc gctcgcgtcg
catccgacac catgctcaga tgcctgatgc gacgctgacg cgtcttatca ggcctaccca
ctgtttttac accgataatt tttccccac ctttttgac tcattcatat aaaaaatata
tttccccacg aaaacgattg ctttttatct tcagatgaat agaatgcggc ggattttttg
ggtttcaaac agcaaaaagg ggaatttc

>3. cvpA/purF(R) 2428784..2429041

tacggtcttg cctgatgcga cgctggcgcg tcttatcagg cctacgcagg ggtagaaccg
taggtcggat aaggcgttta cgccgcaccc gacacgcatt gccgatgcc gcaaaggcat
aaaaagtcga tggcgttgaa tattttttca gcgccatttt tattgatgcg cgggaaggaa
atccctacgc aaacgttttc tttttctgtt agaatgcgcc ccgaacagga tgacagggcg
taaaatcgtg ggacacat

>4. gcvT(R) 3048688..3049134

aaatttctcc tctgttgtt atttgatacc catcacactt tcactctccg gttttttcgc
cgggagattt tctcatttg aaataaacta atttcacctc cgttttcgca ttatattttc
taatgccatt attttttgat ttagtgtttt ttgacatttt ttagctctt aatattgtct
tattcaaatt gactttctca tcacatcatc tttgtataga aactgggtgta ttttttggtt
ttttattctg tcgcgatttt tgcatttttt aaccataagc taatgtgatg atcaatttta
ccttatgggt aacagtctgt ttcggtggtg agttcaggca aaagagaacg attgcgttgg
ggaccgggag tggctccgat gctgggtttc gtggtgataa tttcaccatg aaaaagttgt
cagccccgct tattcaatga ggacaag

>5. glyA(R) 2683528..2683854

atggtcttcc tttttttgca tcttaattga tgtatctcaa atgcatctta taaaaaatag
ccctgcaatg taaatggttc tttggtgttt ttcagaaaga atgtgatgaa gtgaaaaatt
tgcacacaa acctgaaaag aaatccgttt ccggttgcaa gctctttatt ctccaaagcc
ttgcgtagcc tgaaggtaat cgtttgcgta aattcctttg tcaagacctg ttatcgcaca
atgattcggg tatactgttc gccgtgttcc aacaggaccg cctataaagg caaaaaattt
tattgtagc tgagtcagga gatgcgg

>6. guaB(R) 2632091..2632251

gtgagcgaga tcaaattcta aatcagcagg ttattcagtc gatagtaacc cgcccttcgg
ggatagcaag cattttttgc aaaaaggggt agatgcaatc ggttacgctc tgtataatgc
cgcggaata tttattaacc actctggtcg agatattgcc c

>7. purA 4402162..4402264

acaaaaaaca gactgatcga ggtcattttt gagtgcaaaa agtgctgtaa ctctgaaaaa
gcgatggtag aatccatttt taagcaaacg gtgattttga aaa

>8. purC(R) 2595639..2595850

aaatacaggg ctggaatcat ccggcccttt tttctgatat gatacgcaaa cgtgtgcgtc
tgcaggaaaa cgcgatthta gcggtaattc gcacgaaatt tgtttgctcg acgtagttcg
gataaggcgt tcacgccgca tccgacaaaa catccggcac accagacagc aaaagatttt
aaaacgttaa ttcacacca ggagtataaa ag

>9. purE(R) 552324..552440
aaaaccgca actttgctga tttcacagcc acgcaaccgt tttccttgct ctctttccgt
gctattctct gtgccctcta aagccgagag ttgtgcacca caggagtttt aagacgc

>10. purH(R) 4205111..4205724
aaaagtttga tgctcaaaga attaaacttc gtaatgaatt acgtgttcac tcttgagact
tggtattcat ttttcgtctt gcgacgttaa gaatccgtat cttcgagtgc ccacacagat
tgtctgataa attgttaaag agcagttgag acgagcttta gcgactgct gcgaggtggc
gtatattacg ctttcctctt tcagagtcaa ccctgaattt caggatthtt ctcttcaacc
gaaccggctg tttgtgtgaa gtgattcaca tccgccgtgt cgatggaggc gcattatagg
gagttctccg caggccgcaa tagaaaaatt gcagaaaaat gactgactgc tgcattcccc
agcaaaagcc cgcttttatac ctttttacgc acagagttat ccacaatcat caatgtaatt
tctgtatttt gccacggta accacagtca aaattgtgat caccattgaa agagaaaaat
tcgagagcgt tgcgcaaacg ttttcgctac aatgcgggag aaaaataagg atgccccgtt
aggggcgtta gctgagtttt tcgcaaaaaa ttcagctaac gctctctgta atagtcaaat
ccaggggatt tacc

>11. purL(R) 2693564..2693958
acgcaactct ccccgcgctt gaatggcggc gatacggttg tcggctttac caaaccaggg
aatggatggc cagagagcga ccgagagcag cagtgccaga atgccgatga acagataatt
aatctthta tttttcaatt agttaattct ctgtgtctgt cgcgtcccag cttgaaaaaa
cgtaataata gtgaaagggt tactcataaa tgagcggcat tttgcgtaaa cctgcgccag
atggcaactt attacagcca ttggcggcac gcgctgctaa ttcacgatgg tgattttatt
tccacgcaaa cggtttcgct agcgcacag attctttata atgacgcccg tttccccccc
ttgggtacac cgaaagctta gaagacgaga gactt

>12. purM 2618920..2619216
aaaaaaaaatc gacggattat acctcctttc ttcaaggcgg caatattctt ttcgctgact
ttagtcaaaa tgataacggt ttgagataaa gttattttat attcagatgg ttatgaaaga
agattattcc atccgaaaac taacctttac cctggcacia gtcttctttc gccgagcggc
tggggaaaag acgtgcaaaa aggttggtga aagcagtctc gcaaacttt gctttccctg
ttagaattgc gccgaatttt atttttctac cgcaagtaac gcgtggggac ccaagca

>13. purR 1735315..1735867
gccttatctc cacctcttcg cgtcattacg cgatattcat taaagtggcg aaagcatgac
agcaatcaca aaaaaatgaa aataacaaaa agagaaaaca cttttgcat tttgctaaca
aacaggaagg agatgagagg gagaacgcgc tccctcgaga ggaaatcagt gcagcgcggc
agtcaaacc acggctacga tcaaacggag gacgataatc gttgttacca gtgaaaattt
aaggtcggtg ctcacaaagt tttctccttt tttattacca cacaaaaagt gatattacgc
atthtttac actgtgatga aaaaatctcc cgtcatttat aatgataagt gthtttacca
cttccccttt tcgtcaagat cggccaaaat tccacgctta cactatttgc gtactggcca

ttgaccocctt cctgacgctc cgtgctgttt ttccggcgta ccgcaacact tttgttgtgc
gtaaggtgtg taaaggcaaa cgtttacctt gcgattttgc aggagctgaa gttagggctc
ggagtgaaat gga

>14. purT 1928772..1928904

ttgcagcctc tcataataac tgtgatttta tacagtatat ttcttttcgg ttgagaaatc
aacatcagca ataaagacac acgcaaacgt ttcgtttat actgcgcgcg gaattaatca
gggatattc gtt

>15. pyrC(R) 1121831..1121935

tcacgagggc gcattcgcgc cttttatttt tcgtgcaaag gaaaacgttt ccgcttatcc
tttgtgtccg gcaaaaacat cccttcagcc ggagcataga gatta

>16. pyrD 1003881..1003990

aacaggttcg gaaaacgttt gcgttttttt tgccgcaggt caattccctt ttggtccgaa
ctcgcacata atacgcccc ggtttgaca ccgggaatcc aggagagttc

APPENDIX B

PART OF THE SOURCE CODE FOR PCPC AND GS_PC

This appendix lists part of the source code for the algorithms PCPC and GS_PC. The source code for GS_95 and *P*-value calculation is not listed here; they are available online at <http://www.people.fas.harvard.edu/~junliu/index1.html> and <http://ural.wustl.edu/software.html>, respectively as of April 11, 2004.

B.1 Header file

```
#ifndef WC_GIBBS_H
#define WC_GIBBS_H

#include "gibbs.h"

#define LMER_8
#ifdef LMER_6
#define LEN_LMER 6
#endif
#ifdef LMER_7
#define LEN_LMER 7
#endif
#ifdef LMER_8
#define LEN_LMER 8
#endif

typedef struct{
    /* all *-cols in fmodel from site_sampler */
    long *best_cols;
    char *best_word;
    Boolean *best_cols_100;
    /* current 6mer being counted */
```

```

char *cur_6mer;
long *cur_6cols;
Boolean *cur_6cols_100;
long *num_6mers;
long *num_6mers_seq; /* temp. array */
long ***pos_6mers; /* pos_6mers[n][wd_num][num] */
long ***pos_a6mers; /* pos of 6mer actually added to sites */

/* the best 6mer saved */
long *best_6cols;
char *best_6mer;
Boolean *best_6cols_100;
/* p_matrix from Stormo '99 */
char **p_mat_main;
} wc_gibbs_type;

typedef wc_gibbs_type *wgs_type;

static const long t=1; /* ONLY consider type=1 motif */

void comp_prob( long *seq_best_site, ss_type Data, fm_type
                *finalmodel, st_type mapsites );
void adjust_sites( const long *seq_best_site, gs_type G, ss_type
                  Data, fm_type *finalmodel, st_type mapsites );
wgs_type MkGibbs_wc( const ss_type Data, const fm_type M );
void get_best_wordcols( wgs_type wG, ss_type Data, fm_type M );
void get_best_wordcols_PCPC( wgs_type wG );
char GetHighestFreqRes( const double *tfreq, const long *observed_k,
                        const a_type A );
void RunGibbs_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,
                  st_type S );
void SaveBestGibbs_wc( wgs_type wG );
void cnt_best_6mer( wgs_type wG, ss_type Data, const long len_motif );
double tot_alignments( const fm_type M, const st_type S );
double LLR( fm_type M );
double info_wc( fm_type M );
double info_wc_PCPC( wgs_type wG, fm_type M );
void cnt_6mer( wgs_type wG, const ss_type Data, const long len_motif );
Boolean ReservedSite_p6mer( const long ***pos_6mers, const long
                             len_seq, long seq, long len_motif, long pos );
void InitGibbs_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,
                  st_type S );
void InitGibbs_wc_Overlap( gs_type G, wgs_type wG, ss_type Data,
                            fm_type M, st_type S );
void comp_prob_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,

```

```

        st_type S);
void PutGibbs_wc(FILE *fptr, const wgs_type wG, ss_type Data, const
        fm_type M, const long N);
void NilGibbs_wc( wgs_type wG, const long N );

#endif

```

B.2 Source file

```

#include "wc_gibbs.h"

/* remove sites/seqs with low_prob < 0.5.
   ASSUMING:  1. results from sitesamp.c() trustable.
              2. noise seqs not too many.
   **/
void comp_prob( long *seq_best_site, ss_type Data, fm_type
               *finalmodel, st_type mapsites)
{
    long      n, N = NSeqsSeqSet(Data);
    long      len_seq, len_motif, end, s;
    double    best_prob, seq_best_prob, *pos_prob;
    char      *seq; /* numeric format, 1,2,3,4 */

    extern const long t;

    //based on other segments in the final/MAP alignment
    //matrix; best_seg = 1.0 prob.
    for(best_prob=-DBL_MAX, n = 1; n <= N; n++) {
        len_seq = SqLenSeqSet( n, Data );
        len_motif = LenFModel( finalmodel[t] );

        end = len_seq - len_motif + 1;
        pos_prob = PosProbSite(t,n,mapsites);
        seq = SeqSeqSet(n,Data);

        seq_best_prob = -DBL_MAX;
        seq_best_site[n] = 1;
        for(pos_prob[0]=0.0, s = 1; s<= end; s++){
            if(TypeSite(n,s,mapsites) == t){
                //Calc. pos_prob for current site 's'

```

```

        //based on other seg's.
        RmFModel(seq, s, finalmodel[t]);
        pos_prob[s] = (double)LikelihoodFModel(seq, s,
finalmodel[t]);
        Add2FModel(seq,s,finalmodel[t]);
    } else if( len_seq/2 -len_motif +2 <=s && s<= len_seq/2 ){
        //2joint:
        //jump over the middle portion of the jointed seq.
        pos_prob[s] = 0.0;
    } else {
        pos_prob[s] = (double)LikelihoodFModel(seq, s,
finalmodel[t]);
    }

    if( pos_prob[s] > seq_best_prob ){
        seq_best_prob = pos_prob[s];
        seq_best_site[n] = s;
    }
    pos_prob[0] += pos_prob[s]; //no use ??
} //each site.

    best_prob = MAX(double, best_prob, seq_best_prob );
} //each seq.

best_prob = log(best_prob);

//Re-calc pos_prob based on: best_seg prob = 1.
for(n = 1; n <= N; n++) {
    end = SqLenSeqSet(n,Data) - LenFModel(finalmodel[t]) + 1;
    pos_prob = PosProbSite(t,n,mapsites);
    for(s = 1; s <= end; s++) {
        pos_prob[s] = log(pos_prob[s]);
        pos_prob[s] /= best_prob;
        //ratio to best_seg prob in WHOLE seqset.
    }
}
}

/* remove low_prob sites/seqs, and add high_prob sites.
cutoff prob: 0.5.
WARNING: 1. Assuming result comes from site_sampler, i.e. 1
        site/seq.
        2. This function alters G->sites!
**/

```

```

void  adjust_sites( const long *seq_best_site, gs_type G, ss_type Data,
                   fm_type *finalmodel, st_type mapsites )
{
  double  *pos_prob;
  long    site1_pos, n, N= NSeqsSeqSet(Data);

  extern const long  t;

  for(n = 1; n <= N; n++) {
    pos_prob = PosProbSite(t,n,mapsites);
    site1_pos = SitePos( t, n, 1, mapsites );
    //assuming result comes from site_sampler!!
    if(pos_prob[ seq_best_site[n] ] < 0.5){
      //simply remove.
      VacateSite(t,n, site1_pos, mapsites);
      RmFModel(SeqSeqSet(n,Data), site1_pos, finalmodel[t]);
    }
    else if( seq_best_site[n] != site1_pos ){
      //site_1 prob >= 0.5, but not the best.
      //->remove it & add the best site if !occupied.
      VacateSite(t,n, site1_pos, mapsites);
      RmFModel(SeqSeqSet(n,Data), site1_pos, finalmodel[t]);

      if( !OccupiedSite(t,n, seq_best_site[n], mapsites) ){
        //always TRUE: since site_sampler!!
        AddSite(t, n, seq_best_site[n], mapsites);
        Add2FModel( SeqSeqSet(n,Data), seq_best_site[n],
finalmodel[t] );
      }
    }
  }
}

/* create and init a wgs_type object */
wgs_type MkGibbs_wc( const ss_type Data, const fm_type M )
{
  wgs_type wG;
  long      k, n, wd, len_seq, N=NSeqsSeqSet( Data);
  long      ncols=M->ncols, len_motif=M->length;

  NEW(wG, 1, wc_gibbs_type);
}

```

```

/* all *-cols in fmodel from site_sampler */
NEW(wG->best_cols, ncols+2, long);
    //ncols+2: for run_gibbs_wc()!!!
NEW(wG->best_word, ncols+2, char); wG->best_word[ncols+1]= 0;
NEW(wG->best_cols_100, ncols+2, Boolean );

/* current 6mer being counted */
NEW(wG->cur_6mer, LEN_LMER+1, char);
NEW(wG->cur_6cols, LEN_LMER+1, long);
NEW(wG->cur_6cols_100, LEN_LMER+1, Boolean);

NEW(wG->num_6mers, LEN_LMER+2, long);
NEW(wG->num_6mers__seq, LEN_LMER+2, long);

NEWPP( wG->pos_6mers, N+1, long);
NEWPP( wG->pos_a6mers, N+1, long);
for(n=1; n<=N; n++){
    len_seq = SqLenSeqSet(n, Data);
    NEWP( wG->pos_6mers[n], LEN_LMER+2, long);
    NEWP( wG->pos_a6mers[n], LEN_LMER+2, long);
    for( wd=1; wd<=LEN_LMER+1; wd++) {
        //wd=LEN_L_MER+1: the perfect 6mer.
        NEW(wG->pos_6mers[n][wd], len_seq, long);
        NEW(wG->pos_a6mers[n][wd], len_seq, long);
        //!all initialized to 0 upon NEW: marking end.
    }
}

/* the best saved */
NEW(wG->best_6cols, LEN_LMER+1, long);
NEW(wG->best_6mer, LEN_LMER+1, char);
NEW(wG->best_6cols_100, LEN_LMER+1, Boolean);

/* p_matrix from Stormo '99 */
NEWP( wG->p_mat__main, 12, char );
for( k=0; k<12; k++){
    NEW( wG->p_mat__main[k], 60, char );
}
sprintf( wG->p_mat__main[0], "%s", "");
sprintf( wG->p_mat__main[1], "%s", "-L");
sprintf( wG->p_mat__main[3], "%s", "-n");
sprintf( wG->p_mat__main[5], "%s", "-s");
sprintf( wG->p_mat__main[7], "%s", "-A");
sprintf( wG->p_mat__main[8], "%s", "a:t");
sprintf( wG->p_mat__main[9], "%f:%f", M->freq[1], M->freq[4] );
sprintf( wG->p_mat__main[10], "%s", "c:g");

```

```

    sprintf( wG->p_mat__main[11], "%f:%f", M->freq[2], M->freq[3] );

    return wG;
}

/* Coin a word using highest freq. res. of each *-ed col in fmodel
   and record the corresponding col numbers.
**/
void get_best_wordcols( wgs_type wG, ss_type Data, fm_type M )
{
    long j, k, i, *best_cols= wG->best_cols, temp_col;
    char hf_res, *best_word= wG->best_word, temp_res;
    Boolean *best_cols_100= wG->best_cols_100;

    for(best_cols_100[0]=0, k=1, j=M->start; j<=M->end; j++ ){
        if(M->observed[j] != NULL){

            hf_res= GetHighestFreqRes(Data->tfreq, M->observed[j], M->A);
            best_word[k] = hf_res;
            best_cols[k] = k;

            //see if a 100% freq column; if yes,
            //make it required:
            if(TRUE || M->observed[j][hf_res] == M->totsites ){
                best_cols_100[k] = TRUE;
                best_cols_100[0]++;
                //total number of 100% columns!!
            } else {
                best_cols_100[k] = FALSE;
            }

            k++;
        }
    }
    best_word[k] = 0; //mark the end.

    //sort, so that 100% columns are at first.
    for(j=M->ncols+1, i=1; i<=M->ncols; i++){
        if( !best_cols_100[i] && j>i ) j=i;

        if( best_cols_100[i] && j<i ){
            //swap columns i <-> j:
            temp_res = best_word[j]; //save it.
            temp_col = best_cols[j];

```



```

        best_word[j] = best_word[i];
        best_cols[j] = best_cols[i];
        best_cols_100[j] = TRUE;

        best_word[i] = temp_res;
        best_cols[i] = temp_col;
        best_cols_100[i] = FALSE;

        //advance 1.
        j++;
    }
}

return;
}

/* Coin a word using highest freq. res. of each *-ed col in fmodel
   and record the corresponding col numbers.
   **/
void get_best_wordcols_PCPC( wgs_type wG )
{
    long j, *best_cols= wG->best_cols;
    char *best_word= wG->best_word;
    Boolean *best_cols_100= wG->best_cols_100;

    /** CRP pattern parameters **
        const char pttn[]= {0, 1,1,1, 4,3,4,3,1, 0,0,0,0,0,0, 4,2,1,2,1,
                            4,4,4};
        const long ncols= 16, width= 22;
    /***/
    /** LexA pattern parameters **
        const char pttn[]= {0, 4,1, 2,4,3, 4,1,4,1,4,1,4,1,4,1, 2,1,3,
                            4,1};
        const long ncols= 20, width= 20;
    /***/
    /** purR pattern **/
        const char pttn[]= {0, 0,0,0,0,0, 1,2,3,2, 1,1,1,2,3,4,4,4,
                            3,2,3,4, 0,0,0,0,0};
        const long ncols= 16, width= 26;
    /***/

    best_cols_100[0] = ncols;
    for( j=1; j<=width; j++ ){

```

```

        best_word[j] = pttm[j];
        best_cols[j] = j;
        best_cols_100[j] = TRUE;
    }
    best_word[ width+1 ]= 0;    //mark the end.

    return;
}

/* pick and return highest freq. res. in a *-ed column in fmodel.
**/
char  GetHighestFreqRes(const double *tfreq, const long *observed_k,
                        const a_type A)
{
    char      b, hf_res;
    double    w_freq, best_wfreq; //w_: weighted.
    //const double  FREQ_CUTOFF =0.50;

    hf_res =1;
    best_wfreq = (double)observed_k[1]/tfreq[1];
    for( b=2; b<=nAlpha( A ); b++ ){
        if( ( w_freq=(double)observed_k[b]/tfreq[b] ) > best_wfreq ){
            //weighted freq; will favor low_numbered res. when tie.
            best_wfreq =w_freq;
            hf_res =b;
        }
    }

    return hf_res;
}

void  RunGibbs_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,
                  st_type S)
{
    long      ncols_100, start, end, i[LEN_LMER+1], j;
    long      ncols=M->ncols, len_motif=M->length;
    char      *best_word= wG->best_word, *cur_6mer= wG->cur_6mer;
    long      *best_cols= wG->best_cols, *cur_6cols= wG->cur_6cols;
    Boolean   *best_cols_100= wG->best_cols_100,
              *cur_6cols_100= wG->cur_6cols_100;
    double    info, info_PCPC, tot_info, best_info, best_tot_info;
    double    score, best_score; //-Log-Likelihood ratio, Stormo[99].

```

```

Double    best_info_zscore, info_zscore;
Double    tot_info_zscore, best_tot_info_zscore;
long      count;
mh_type   wH;

double    p_value, lowest_p_value;
double    tot_aligns, expect_freq, lowest_expect_freq;
char      **p_mat__main= wG->p_mat__main ;
double    z_score, best_z_score;
long      c0, c1, k;
double    c0_exp, ratio, best_ratio=0.0;
FILE      *fp_out;

    if((fp_out=fopen("c:\\Documents and Settings\\
\\Liang \\My Documents\\Gibbs9_95_H\\GS_PCPC_results\\ .doc",
"a+")) == NULL){
        fprintf(stderr, "Can't open file for writing.\n");
        exit(1);
    }

    wH = Mheap(100, 3);
    //get_best_wordcols( wG, Data, M);
    get_best_wordcols_PCPC( wG );

    best_info = 0.0;
    best_tot_info = 0.0;
    best_score = -DBL_MAX;
    lowest_p_value = DBL_MAX;
    lowest_expect_freq = DBL_MAX;
    best_z_score = 0.0;
    best_info_zscore = 0.0;
    best_tot_info_zscore = 0.0;
    count = 0;

    //now, check for 100% columns. If there are >=6 of this
    //kind, then only count them. Otherwise, use thos <6 100%
    //columns as required, and complete the 6mer with ~100%
    //columns in counting.
    ncols_100 = best_cols_100[0];
    if( ncols_100 >= LEN_LMER ){
        start =1; //for inside loop below.
        end = M->ncols;
    } else {
        fprintf( stderr, "\n\nPCPC WARNING: ncols_100 < LEN_LER!" );
        fprintf( stderr, "\nExited 1.\n" );
    }

```

```

exit(11);

start = ncols_100 +1;
end = ncols;

for(k=1; k<start; k++){
    cur_6mer[k]= best_word[ k ];
    cur_6cols[k]= best_cols[ k ];
    cur_6cols_100[k]= best_cols_100[ k ];
}
}

for(i[1]=1; i[1]<=end-LEN_LMER+1; i[1]++){
    if( best_word[ i[1] ] == 0 ) continue;
    for(i[2]=i[1]+1; i[2]<=end-LEN_LMER+2; i[2]++){
        if( best_word[ i[2] ] == 0 ) continue;
        for(i[3]=i[2]+1; i[3]<=end-LEN_LMER+3; i[3]++){
            if( best_word[ i[3] ] == 0 ) continue;
            for(i[4]=i[3]+1; i[4]<=end-LEN_LMER+4; i[4]++){
                if( best_word[ i[4] ] == 0 ) continue;
                for(i[5]=i[4]+1; i[5]<=end-LEN_LMER+5; i[5]++){
                    if( best_word[ i[5] ] == 0 ) continue;
#if defined(LMER_6) || defined(LMER_7) || defined(LMER_8) ||
defined(LMER_9)
                    for(i[6]=i[5]+1; i[6]<=end-LEN_LMER+6; i[6]++){
                        if( best_word[ i[6] ] == 0 ) continue;
#endif
#if defined(LMER_7) || defined(LMER_8) || defined(LMER_9)
                    for(i[7]=i[6]+1; i[7]<=end-LEN_LMER+7; i[7]++){
                        if( best_word[ i[7] ] == 0 ) continue;
#endif
#if defined(LMER_8) || defined(LMER_9)
                    for(i[8]=i[7]+1; i[8]<=end-LEN_LMER+8; i[8]++){
                        if( best_word[ i[8] ] == 0 ) continue;
#endif
#if defined(LMER_9)
                    for(i[9]=i[8]+1; i[9]<=end-LEN_LMER+9; i[9]++){
                        if( best_word[ i[9] ] == 0 ) continue;
#endif
                    {
                        //check for 100% freq columns; they are
                        //required and will always be used in counting.
                        //-> this will be entered only if 0< ncols_100< LEN_LER.
                        if(i[1]!=ncols+1 && 1<=ncols_100 && ncols_100<LEN_LMER){
                            //then is first time reaching here, and there are
                            //[1,5] 100% freq cols.

```

```

        fprintf( stderr, "\n\nPCPC WARNING: ncols_100 <
LEN_LER!" );

        fprintf( stderr, "\nExited 2.\n" );
        exit(22);

        for(j=1; j<start; j++){
            i[j]= ncols+1;
            //this way, they won't be executed again.
            //->not really needed to be: ncols+1;
            // large enough is fine.
        }
    }

    //get current 6mer and 6cols.
    for(j=start; j<=LEN_LMER; j++){
        cur_6mer[j]= best_word[ i[j] ];
        cur_6cols[j]= best_cols[ i[j] ];
        cur_6cols_100[j]= best_cols_100[ i[j] ];
    }
    cnt_6mer(wG, Data, len_motif);
    InitGibbs_wc( G, wG, Data, M, S );

    //see if <=3 sites found; yes, go to next 6mer.
    // -> if 0, then P_matrix_main() will always fail !!
    //     if 1, and tot_info > -21.0, then also fail !!
    //     and so on.
    //     e.g., if tot_info ~ 200.0, then all M->totsites
    //     <= LEN_L_MER+1 will always fail !
    //So, skip them!
    //=>7: just an emprical shreshold:
    if(M->totsites < 7 ){
        continue;
    }

    //calc. c0, c0/c1 - from Hampton (UCI).
    c0=wG->num_6mers[LEN_LMER+1];
    c1=wG->num_6mers[0]-c0;
    for(c0_exp=1.0, k=1; k<=LEN_LMER; k++){
        c0_exp *= Data->tfreq[ cur_6mer[k] ];
    }
    ratio=(double)c0/c0_exp;

    //calc. z-score, from Tompa (UW).
    z_score = z__score( wG, M, S );

```

```

//calc. info content.
score = -LLR( M );
info = info_wc( M ); //all-cols.
//info = info_wc_PCPC( wG, M ); //nt-cols.
tot_info = info * (double)M->totsites;
//tot_info: N*I_seq!!

//calc. p-value, from Stormo '99.
sprintf( p_mat__main[2], "%ld", M->ncols ); //all-cols.
//sprintf( p_mat__main[2], "%ld", wG->best_cols_100[0] );
sprintf( p_mat__main[4], "%ld", M->totsites );
sprintf( p_mat__main[6], "%.3f", tot_info );

/** Debug **
fp_out = fopen("OUT_p_value.doc", "a");
fprintf(fp_out, "%2d-%2d-%2d-%2d-%2d-%2d\n",
i[1],i[2],i[3],i[4],i[5],i[6]);
fprintf(fp_out, "%4ld, %6.2f", M->totsites, tot_info );
fprintf(fp_out, "\n\n");
fclose(fp_out);
****/

//calc. p-value, from Stormo '99.
p_value = P_matrix__main( 12, p_mat__main );
tot_aligns = tot_alignments( M, S );
expect_freq = tot_aligns * p_value;

//combining 2 criteria:
info_zscore = info *z_score;
tot_info_zscore = tot_info *z_score;

InsertMheap( (keytyp)(-expect_freq), wH);

//criterion here:
//if( info_zscore > best_info_zscore ){
//if( tot_info_zscore > best_tot_info_zscore ){
//if( p_value < lowest_p_value ){

if( tot_info > best_tot_info ){
//if( z_score > best_z_score ){
//if( expect_freq < lowest_expect_freq ){
++count;

best_ratio=ratio;
best_info_zscore = info_zscore;
best_tot_info_zscore = tot_info_zscore;
best_info = info;

```

```

        best_score = score;
        best_z_score = z_score;
        best_tot_info = tot_info;
        lowest_p_value = p_value;

        lowest_expect_freq = expect_freq;

        SaveBestGibbs_wc( wG );

/** Debug **/
fprintf(stderr,
"\n\n=====\n");
fprintf(stderr, "\n\t # %ld\n", count);
fprintf(stderr, "\n\t score:      %.2f", score);
fprintf(stderr, "\n\t info:      %.2f", info );
fprintf(stderr, "\n\t tot_info:   %.2f", tot_info );
fprintf(stderr, "\n\t p_val:      %g", p_value );
fprintf(stderr, "\n\t expect_freq: %g", expect_freq );
fprintf(stderr, "\n\t z_score:   %.2f", z_score );
fprintf(stderr, "\n\t info_z:    %.2f", info_zscore );
fprintf(stderr, "\n\t tot_info_z: %.2f", tot_info_zscore );
fprintf(stderr, "\n\t nsites:   %ld\n", M->totsites);

PutGibbs_wc( stderr, wG, Data, M, NSeqsSeqSet(Data));
fprintf(stderr, "\n");
****/

/** Debug - crp **
if(cur_6mer[1]== 4 && cur_6mer[2]== 3 && cur_6mer[3]== 4
    && cur_6mer[4]== 3 && cur_6mer[5]== 1 && cur_6mer[6]== 2){
    fprintf(fp_out, "\n\nTGTGA.....C\n");
}
****/

/** Debug - LexA **
if(cur_6mer[1]== 2 && cur_6mer[2]== 4 && cur_6mer[3]== 3
    && cur_6mer[4]== 1 && cur_6mer[5]== 2 && cur_6mer[6]== 1){
    fprintf(fp_out, "\n\nCTG[1]A[8]CA\n");
}
****/

/** Debug - purR **
if(cur_6mer[1]== 3 && cur_6mer[2]== 1 && cur_6mer[3]== 2
    && cur_6mer[4]== 3 && cur_6mer[5]== 4 && cur_6mer[6]== 2){
    fprintf(fp_out, "\n\nG[2]A[1]CG[2]T[1]C\n");
}

```

```

}
/****/

//goto EXIT;

        } //if criterion.
    } //for all 6mer.

        }
    }
}

#if defined(LMER_6) || defined(LMER_7) || defined(LMER_8) ||
defined(LMER_9)
    }
#endif
#if defined(LMER_7) || defined(LMER_8) || defined(LMER_9)
    }
#endif
#if defined(LMER_8) || defined(LMER_9)
    }
#endif
#if defined(LMER_9)
    }
#endif

//EXIT:

/* now re-count the best_6mer saved and output the result */
fprintf(fp_out, "\n\n\n=====");
fprintf(fp_out, "\npurR, tot_info, 8-pttn: \n");
fprintf(fp_out, "\n\t expect_freq: %g", lowest_expect_freq );
fprintf(fp_out, "\n\t z_score:      %.2f", best_z_score );
fprintf(fp_out, "\n\t tot_info:      %.2f", best_tot_info );

cnt_best_6mer( wG, Data, len_motif);
InitGibbs_wc( G, wG, Data, M, S);
comp_prob_wc( G, wG, Data, M, S);
PutGibbs_wc( fp_out, wG, Data, M, NSeqsSeqSet(Data));

/* output best wc_results */
if(G->fragment) {
    NullSitesFModel(G->>null, M);
}

```



```

        PutSites(fp_out, t, S, ProbSite(t, S),G->null);
    } else {
        PutSites(fp_out, t, S, ProbSite(t, S),NULL);
    }
    PutFModel(stderr, M);

    fclose(fp_out);
    return;
} /* The END */

/* count all exact and 1-mismatch 6mers */
void cnt_6mer( wgs_type wG, const ss_type Data, const long len_motif )
{
    long n, N= NSeqsSeqSet(Data), end;
    char r, *seq, *cur_6mer= wG->cur_6mer;
    long *cur_6cols= wG->cur_6cols, ***pos_6mers= wG->pos_6mers;
    long *num_6mers= wG->num_6mers, *num_6mers__seq=wG->num_6mers__seq;
    long s, wd, pos, i;
    long first_col, len_seq, site_f;
    Boolean *cur_6cols_100= wG->cur_6cols_100;

    /* Debug--CRP sites **
    cur_6mer[1]= 4; cur_6mer[2]= 4; cur_6mer[3]= 2; cur_6mer[4]= 1;
    cur_6mer[5]= 2;
    //cur_6mer[6]= 2;
    //cur_6mer[7]= 1;
    //cur_6mer[8]= 1;

    cur_6cols[1]= 4; cur_6cols[2]= 15; cur_6cols[3]= 16; cur_6cols[4]= 17;
    cur_6cols[5]= 18;
    //cur_6cols[6]= 13;
    //cur_6cols[7]= 14;
    //cur_6cols[8]= 16;
    /***/

    /* Debug--LexA sites **
    cur_6mer[1]= 2; cur_6mer[2]=4; cur_6mer[3]= 3; cur_6mer[4]= 1;
    cur_6mer[5]= 2; cur_6mer[6]= 1;
    cur_6cols[1]= 1; cur_6cols[2]= 2; cur_6cols[3]= 3; cur_6cols[4]= 5;
    cur_6cols[5]= 14; cur_6cols[6]= 15;
    /***/

    /* Debug--purR sites **
    cur_6mer[1]= 3; cur_6mer[2]= 1; cur_6mer[3]= 2; cur_6mer[4]= 3;

```

```

cur_6mer[5]= 4; cur_6mer[6]= 2;
cur_6cols[1]= 1; cur_6cols[2]= 4; cur_6cols[3]= 6; cur_6cols[4]=
LEN_L_MER+1;
cur_6cols[5]= 10; cur_6cols[6]= 12;
/****/

/* first clear previous results */
for(wd=0; wd<=LEN_LMER+1; wd++) num_6mers[wd]= 0;

/* find lowest col_num for cur_6mer.
   REASON: due to that 100% cols were sorted to first cols,
           lowest col_num may not be the 1st col.
   **/
first_col = cur_6cols[1];
for( pos=2; pos<=LEN_LMER; pos++ ){
    if( cur_6cols[pos] < first_col )
        first_col = cur_6cols[pos];
}

for( n =1; n<= N; n++){
    //for each seq.
    len_seq = SqLenSeqSet( n, Data );
    end = len_seq -len_motif +1;
    //2joint:
    //the middle portion will be taken care of below.

    for( wd=1; wd<=LEN_LMER+1; wd++) num_6mers__seq[wd]=0;

    seq = SeqSeqSet( n, Data );
    /* now scan all possible pos on current seq */
    for( s=first_col; s<=end+ (first_col-1); ){
        //WARNING: first_col number may not be 1.

        //2joint:
        //jump over the end portion of the original seq.
        if( len_seq/2 -len_motif +2 +(first_col-1) <=s
            && s<= len_seq/2 +(first_col-1) ){
            s++;
            continue;
        }
        //2joint.
        //get the right starting position of the motif instance.
        site_f = s - (first_col -1);
    }
}

```

```

for( pos=1; pos<=LEN_LMER; pos++){ /* check all 6mers */
  r= seq[s+ cur_6cols[pos]- first_col];
  switch( r == cur_6mer[pos] ){
  case FALSE:
    //first see if this is a 100% column, i.e.
    //if this is a required res.
    if( !cur_6cols_100[pos] ){
      //then not a required pos:
      /* then must be a 1-mismatcher, or nothing */
      for( i=pos+1; i<=LEN_LMER; i++ ){
        r= seq[s+ cur_6cols[i]- first_col];
        if( r != cur_6mer[i]){
          /* mismatch again; nothing: */
          break;
        }
      }
      if( i == LEN_LMER+1 ){
        /* then one of the 1-mismatch 6mers */
        num_6mers[pos]++; num_6mers[0]++;
        num_6mers__seq[pos]++;
        pos_6mers[n][pos][num_6mers__seq[pos]] = site_f;
      }
    }
    s += 1;
    pos = LEN_LMER+2; /* pos>LEN_LMER+1 always ok */
    break;

  case TRUE:
    break; /*nop.
  }
} /* for pos */

/* is it a perfect match 6mer? */
if( pos == LEN_LMER+1 ){
  num_6mers[ LEN_LMER+1]++; num_6mers[0]++;
  num_6mers__seq[ LEN_LMER+1]++;
  pos_6mers[n][LEN_LMER+1][ num_6mers__seq[LEN_LMER+1] ] =
site_f;

  s += 1; //overlapping-counting.
}
} /* for s */

for( wd=1; wd<=LEN_LMER+1; wd++){

```

```

        pos_6mers[n][wd][ ++num_6mers__seq[wd] ] =0;
        //mark the end.
    }
} /* for n */
}

void SaveBestGibbs_wc( wgs_type wG )
{
    long i;
    for(i=1; i<=LEN_LMER; i++){
        wG->best_6mer[i] = wG->cur_6mer[i];
        wG->best_6cols[i] = wG->cur_6cols[i];
        wG->best_6cols_100[i] = wG->cur_6cols_100[i];
    }
}

void cnt_best_6mer( wgs_type wG, ss_type Data, const long len_motif)
{
    /* first restore the best 6mer saved, then count */
    long i;
    for(i=1; i<=LEN_LMER; i++){
        wG->cur_6mer[i] = wG->best_6mer[i];
        wG->cur_6cols[i] = wG->best_6cols[i];
        wG->cur_6cols_100[i] = wG->best_6cols_100[i];
    }
    cnt_6mer( wG, Data, len_motif);
}

/* counting total possible number of alignments, given current
sequence data and width of motif which is always used irrespective
of actual k-pattern width.
Based on the paper of Stormo '99.
NOTE: return double instead of long due to possibility of overflow.
**/
double tot_alignments( const fm_type M, const st_type S )
{
    long n, len_elem= S->len_elem[t], nwords= M->totsites;
    double tot_aligns;
    static long N=0; //make N static so as only to compute once.

    //N calculated once and for all, since static.
    if( N == 0 ){
        //tot possible starting points for the motif/k-pattern.

```

```

    for( n=1; n<=NSeqsSeqSet(S->data); n++ )
        N += S->len_seq[n]/2 - len_elem + 1;
        //counting ONE strand, since the rc_strand is not
        //random; it is related to forw_strand!!
    }

    tot_aligns = lgamma(N+1.0) - lgamma(N-nwords+1.0) -
lgamma(nwords+1.0);
        //WARNING: have to add 1.0
        //          -> lgamma(3.0) = fact(2)!

    return exp(tot_aligns);
}

/* calc. info content for fmodel. */
double info_wc( fm_type M )
{
    long b, j;
    double p, q, ib, info, total, tot_info;

    if(M->update) update_fmodel(M);
    for(tot_info = 0.0, j=M->start; j<= M->end; j++){
        if(M->observed[j]!=NULL){
            //get total res count for current col.
            for(total=0.0, b = 1; b <= nAlpha(M->A); b++){
                total += (double) M->observed[j][b] + M->Ps[b];
            }
            //calc. IC for current col.
            for(info=0.0, b = 1; b <= nAlpha(M->A); b++){
                p = ( (double)M->observed[j][b] + M->Ps[b] ) / total;
                if(p > 0.0){
                    q = M->freq[b];
                    //WARNING: should update since seq_rc was counted!!
                    ib = p*log(p/q)/log(2.0);
                    info += ib;
                }
            }
            tot_info += info;
        }
    }

    return tot_info;
}

```

```

/* calc. info content for fmodel.
   NOTE: only sum those non-0 columns.
   **/
double  info_wc_PCPC( wgs_type wG, fm_type M )
{
    long  b, j, j2;
    double p, q, ib, info, total, tot_info;
    char  *best_word= wG->best_word;

    if(M->update) update_fmodel(M);
    for(tot_info=0.0, j2=1, j= M->start; j<= M->end; j++, j2++){
        if( best_word[j2] != 0 ){
            //get total res count for current col.
            for(total=0.0, b = 1; b <= nAlpha(M->A); b++){
                total += (double) M->observed[j][b] + M->Ps[b];
            }
            //calc. IC for current col.
            for(info=0.0, b = 1; b <= nAlpha(M->A); b++){
                p = ( (double)M->observed[j][b] + M->Ps[b] ) / total;
                if(p > 0.0){
                    q = M->freq[b];
                    //WARNING: should update since seq_rc was counted!!
                    ib = p*log(p/q)/log(2.0);
                    info += ib;
                }
            }
            tot_info += info;
        }
    }

    return tot_info;
}

/* the P value (prob.) of an individual alignment matrix as determined
   by the multi-nomial distri., given the assumption that the distri.
   of letters is independent and identically distributed.
   Referring to:
   Hertz GZ and Stormo GD (1999) Identifying DNA and protein patterns
   with statistically significant alignments of multiple sequences,
   Bioinformatics, 15, 563-577.
   **/
double  LLR( fm_type M )
{
    long  b, j;

```

```

double LLR, LLR_j, N_segs, num_b, freq_b, term2;

if(M->update) update_fmodel(M);

N_segs = (double)M->totsites + M->npseudo;
for(LLR=0.0, j=M->start; j<= M->end; j++){
    if(M->observed[j] != NULL){
        //then *-ed column in fmodel; being used.
        for(LLR_j=0.0, b = 1; b <= nAlpha(M->A); b++){
            if( M->Ps[b] != 0.0) {
                //then res. 'b' is present in the dataset.
                num_b = (double)M->observed[j][b] + M->Ps[b];
                freq_b = num_b / N_segs;
                term2 = log( M->freq[b] / freq_b);

                LLR_j += num_b* term2;
            }
        }
        LLR += LLR_j;
    }
}

return LLR;
}

/* Assuming pos is given in forward notation;
   if negative, then counting backward and '-pos' marks the END of the
   word.
**/
Boolean ReservedSite_p6mer(const long ***pos_6mers, const long len_seq,
                           long seq, long len_motif, long pos)
{
    long num, pos_p6mer;
    const long CRP_Stormo=0; //base on Stormo[99].

    for(num=1; (pos_p6mer=pos_6mers[seq][ LEN_LMER+1 ][num]) != 0;
num++){
        if(pos_p6mer > len_seq/2){
            //then: in 2nd half.
            pos_p6mer = (len_seq+1) -pos_p6mer -len_motif +1;
        }

        if( abs(pos_p6mer -pos) < len_motif+ CRP_Stormo)

```

```

        return TRUE;
        //+CRP_Stormo: based on Stormo[99] !!
    }

    return FALSE;
}

void InitGibbs_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,
                  st_type S)
{
    long j, b, n, N=NSeqsSeqSet(Data), k, end, s, s2, s_f;
    long wd, num, num_a, len_seq;
    long len_motif=M->length;
    long ***pos_6mers=wG->pos_6mers, ***pos_a6mers= wG->pos_a6mers;
    long *tmp_pos=G->pos;

    extern const long t;

    /* Debug **
    pos_6mers[1][5][1]=0;
    pos_6mers[3][4][1]=0;
    pos_6mers[LEN_L_MER+1][1][1]= -60; pos_6mers[LEN_L_MER+1][1][2]= 0;
    pos_6mers[8][1][1]=0;
    pos_6mers[9][6][1]=0;
    pos_6mers[17][3][1]=0;
    pos_6mers[17][6][1]=0;
    /***/

    /* first, clear original fmodel completely.
       reason: hard to know sites' directions, f_ or rc_ ?
    */
    for( j=M->start; j<=M->end; j++ ){
        if(M->observed[j] != NULL){
            for( b = 1; b <= nAlpha(M->A); b++ ){
                M->observed[j][b] = 0; //thing_1: to be worried.
            }
        }
    }
    M->totsites = 0; //thing_2.

    for(n = 1; n <= N; n++) {

```



```

/* thing_3: remove original sites on the seq */
PostTSites(t,n, tmp_pos, S);
end = nSites(t,n, S );
for(k = 1; k <= end; k++){
    s = tmp_pos[k];
    VacateSite(t,n,s, S );
}

/* now, add wc_gibbs sites to the seq;
   perfect 6mer site are guaranteed to be added, while 1-mismatch
   6mer site are subject to confliction check before being added.
**/
len_seq = SqLenSeqSet(n, Data);
for(wd=1; wd<=LEN_LMER+1; wd++){
    //wd==LEN_LMER+1: the perfect match 6mer.
    for(num_a=num=1; (s= pos_6mers[n][wd][num]) != 0; num++){
        /* first determine if 's' is a reserved site for p6mers, or
           occupied site by some just added 1-mismatch 6mer(s).
           NOTE: site for a perfect 6mers will always be available,
               i.e., won't be occupied.
        **/

/** Debug - Overlapping sites allowed: **/

        s2 = (len_seq+1)- s- S->len_elem[t] +1;
        //e.g. 211- 106 -17+1 = 105-16= 89: the staring pos, OR
        //      211- 89 -17+1 = 122-16= 106.
        s_f = (s<=len_seq/2) ? s : s2;

        //see if imperfect 6mer; if yes, do not add it if:
        //1> reserved site for p6mer;
        //2> already occupied site by another imperfect 6mer.
        if( wd !=LEN_LMER+1 &&
            ( ReservedSite_p6mer(pos_6mers, len_seq, n,
len_motif, s_f)
            || OccupiedSite(t,n, s, S) || OccupiedSite(t,n,
s2,S) )){
            continue;
        }

        //see if p6mer.
        if( wd == LEN_LMER+1 && (OccupiedSite(t,n, s, S) ||
OccupiedSite(t,n, s2,S) )){
            //only could be occupied by another p6mer.
            continue;
        }

```

```

/****/

        AddSite(t,n, s, S);
        Add2FModel(SeqSeqSet(n,Data), s, M);
        pos_a6mers[n][wd][ num_a++ ] = s; //luckily added site.

    } /* for num */

        pos_a6mers[n][wd][ num_a ] = 0; //mark the END.
    } /* for wd */
} /* for n */

/* thing_4: re-compute pseudo res. counts */
M->npseudo = M->totsites * G->pseudo;
for(b = 1; b <= nAlpha(G->A); b++){
    M->Ps[b] = M->npseudo * M->freq[b];
    //WARNING: M->freq[b] not updated for seqset_rc!!
}
} /* end */

void InitGibbs_wc_Overlap(gs_type G, wgs_type wG, ss_type Data,
                        fm_type M, st_type S)
{
    long j, b, n, N=NSeqsSeqSet(Data), k, end, s, s2, s_f;
    long wd, num, num_a, len_seq;
    long len_motif=M->length;
    long ***pos_6mers=wG->pos_6mers, ***pos_a6mers= wG->pos_a6mers;
    long *tmp_pos=G->pos;
    static Boolean is_first_time_here = TRUE;
        //then sites on seqs added by AddSite() in Gibbs sampler,
        //not by AddSite_wc_overlap() in wc_gibbs.

    extern const long t;

    /* first, clear original fmodel completely.
       reason: hard to know sites' directions, f_ or rc_.
    **/
    for( j=M->start; j<=M->end; j++ ){
        if(M->observed[j] != NULL){
            for( b = 1; b <= nAlpha(M->A); b++ ){
                M->observed[j][b] = 0; //thing_1: to be worried.
            }
        }
    }
}

```

```

M->totsites = 0; //thing_2.

for(n = 1; n <= N; n++) {
  /* thing_3: remove original sites on the seq */
  PostSites(t,n, tmp_pos, S);
  end = nSites(t,n, S );
  for(k = 1; k <= end; k++){
    s = tmp_pos[k];
    if( is_first_time_here ){
      //then, need to remove sites added by AddSite():
      VacateSite( t,n,s, S );
    } else {
      //then, remove sites added by AddSite_wc_overlap():
      VacateSite_wc_overlap(t,n,s, S );
    }
  }
}

/* now, add wc_gibbs sites to the seq;
   perfect 6mer site are guaranteed to be added, while 1-mismatch
   6mer site are subject to confliction check before being added.
**/
for(wd=1; wd<=LEN_LMER+1; wd++){
  for(num_a=num=1; (s= pos_6mers[n][wd][num]) != 0; num++){
    AddSite_wc_overlap(t,n, s, S); //Overlap sites allowed!
    Add2FModel(SeqSeqSet(n,Data), s, M);

    pos_a6mers[n][wd][ num_a++ ] = s; //luckily added site.
  } /* for num */

  pos_a6mers[n][wd][ num_a ] = 0; //mark the END.
} /* for wd */
} /* for n */

/* thing_4: re-compute pseudo res. counts */
M->npseudo = M->totsites * G->pseudo;
for(b = 1; b <= nAlpha(G->A); b++){
  M->Ps[b] = M->npseudo * M->freq[b];
  //WARNING: M->freq[b] not updated for seqset_rc!!
}

is_first_time_here = FALSE;
//from now on, will be using VacateSite_wc_overlap()!!

} /* end */

```

```

void comp_prob_wc( gs_type G, wgs_type wG, ss_type Data, fm_type M,
                  st_type S)
{
    long      len_seq, n, N= NSeqsSeqSet(Data);
    double    seq_best_prob;
    long      ***pos_a6mers= wG->pos_a6mers;
    long      len_motif= M->length, k, end;
    double    best_prob, *pos_prob;
    char      *seq, *seq_rc;      /* numeric format, 1,2,3,4 */
    long      wd, num_a, s, ss, s_f, s_rc;

    extern const long t;

    //based on other segments in the final/MAP alignment matrix;
    //best_seg = 1.0 prob.
    best_prob= -DBL_MAX;
    for( n = 1; n <= N; n++) {
        len_seq= SLenSeqSet(n, Data);
        seq_best_prob= -DBL_MAX;

        pos_prob= PosProbSite(t, n, S);
        PostSites(t,n, G->pos, S);
        end = nSites(t,n, S );

        seq = SeqSeqSet(n,Data);
        for(k = 1; k <= end; k++){
            s = G->pos[k];

            RmFModel(seq, s, M);
            pos_prob[s] = (double)LikelihoodFModel(seq, s, M);
            Add2FModel(seq,s,M);

            if( pos_prob[s] > seq_best_prob ){
                seq_best_prob = pos_prob[s];
            }
        } //for k.
        best_prob = MAX(double, best_prob, seq_best_prob );
    }

    //scale pos_prob based on: best_prob=1.00.
    best_prob = log(best_prob);
    for(n = 1; n <= N; n++) {
        pos_prob= PosProbSite(t, n, S);

```

```

    PostTSites(t,n, G->pos, S);
    end = nSites(t,n, S );
    for(k = 1; k <= end; k++){
        s = G->pos[k];
        pos_prob[s] = log(pos_prob[s]);
        pos_prob[s] /= best_prob;
    }
}

} /* end */

void PutGibbs_wc(FILE *fptr, const wgs_type wG, ss_type Data,
                const fm_type M, const long N)
{
    long      n, i, wd, num, s, s_f, len_seq, len_elem=M->length;
    a_type    A= M->A;

    fprintf(fptr, "\n\nNow outputs wc_gibbs result for BEST 6mer and
6cols:\n\n");

    /* the best_cols and word */
    fprintf(fptr, "\nBest cols: ");
    for(i=1; i<=len_elem; i++)
        fprintf(fptr, "%3ld", wG->best_cols[i]);

    fprintf(fptr, "\nBest word: ");
    for(i=1; i<=len_elem; i++) {
        if(wG->best_word[i] == 0)
            fprintf(fptr, " .");
        else
            fprintf(fptr, "%3c", AlphaChar(wG->best_word[i], A) );
    }

    /* the best_6cols and 6mer */
    fprintf(fptr, "\n\nBest 6cols: ");
    for(i=1; i<=LEN_LMER; i++) fprintf(fptr, "%4ld", wG->cur_6cols[i]);
    fprintf(fptr, "\nBest 6mer: ");
    for(i=1; i<=LEN_LMER; i++)
        fprintf(fptr, "%4c", AlphaChar(wG->cur_6mer[i], A) );

    /* all best 6mers and their positions in seqset */
    fprintf(fptr, "\n\nAll best 6mers and their positions:\n");
    for(n=1; n<=N; n++){
        fprintf(fptr, "\n seq %3ld: ", n);

```

```

len_seq = SqLenSeqSet(n, Data);
for(wd=1; wd<=LEN_LMER+1; wd++){
    for(num=1; (s= wG->pos_a6mers[n][wd][num]) != 0; num++){
        if(s<= len_seq/2) s_f = s;
        else s_f = -( (len_seq+1) - s -len_elem +1 );

        fprintf(fptr, "%8ld[ %ld]", s_f, wd);
    }
}
}
fprintf(fptr, "\n\n Added sites: %ld", M->totsites );
fprintf(fptr, "\n Total sites: %ld\n\n", wG->num_6mers[0] );
}

```

```

void NilGibbs_wc( wgs_type wG, const long N )
{
    long k, n, wd;

    /* all *-cols in fmodel from site_sampler */
    free( wG->best_cols );
    free( wG->best_word );
    free( wG->best_cols_100 );

    /* current 6mer being counted */
    free( wG->cur_6mer);
    free( wG->cur_6cols);
    free( wG->cur_6cols_100 );

    free( wG->num_6mers);
    free( wG->num_6mers__seq);

    for(n=1; n<=N; n++){
        for( wd=1; wd<=LEN_LMER+1; wd++){
            free( wG->pos_6mers[n][wd]);
            free( wG->pos_a6mers[n][wd]);
        }
        free( wG->pos_6mers[n]);
        free( wG->pos_a6mers[n]);
    }
    free( wG->pos_6mers);
    free( wG->pos_a6mers);

    /* the best saved */
    free( wG->best_6cols);
}

```

```
free( wG->best_6mer);
free( wG->best_6cols_100 );

/* p_matrix from '99 */
for( k=0; k<12; k++ ){
    free( wG->p_mat__main[k] );
}
free( wG->p_mat__main );

free( wG);
}
```

VITA 

Liang Zhao

Candidate for the Degree of

Master of Science

Thesis: IDENTIFICATION OF TRANSCRIPTION FACTOR BINDING SITES

Major Field: Computer Science

Biographical:

Personal Data: Born in Zibo, Shandong, China, on September 12, 1969, the son of Yongde Zhao and Yulan Xiang.

Education: Graduated from Zibo No. 5 High School, Shandong, China in June 1988; received Bachelor of Science degree in Chemistry from Zhejiang University, Hangzhou, China in July 1992; received Master of Engineering degree in Polymer Science and Engineering from Beijing Research Institute of Chemical Industry, Beijing, China in August 1995. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May 2004.

Experience: Raised in a small town in Zibo, Shandong, China; employed by Polyolefins National Engineering and Research Center, Beijing, China as a research associate, September 1995 to March 1999; employed by Computer Science Department, Oklahoma State University as a graduate teaching assistant, January 2001 to December 2003.