STOCHASTIC HEURISTIC PROGRAM FOR TARGET MOTIF IDENTIFICATION

STOCHASTIC HEURISTIC PROGRAM FOR TARGET MOTIF IDENTIFICATION

By

XIAN ZHANG, B.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (1999) MCMASTER UNIVERSITY

(Computer Science) Hamilton, Ontario

TITLE: Stochastic Heuristic Program for Target Motif Identification

AUTHOR: Xian Zhang

B.Sc. (Heilongjiang Aug. 1 University, Mishan, China)

SUPERVISOR: Dr. Tao Jiang

NUMBER OF PAGES: x, 109

Abstract

Identifying motifs that are "close" to one or more substrings in each sequence in a given set of sequences and hence characterize that set is an important problem in computational biology. The target motif identification problem requires motifs that characterize one given set of sequences but are far from every substring in another given set of sequences. This problem is NP-hard and hence is unlikely to have efficient optimal solution algorithms. In this thesis, we propose a set of modifications to one of the most popular stochastic heuristics for finding motifs, Gibbs Sampling [LAB+93], which allow this heuristic to detect target motifs. We also present the results of four simulation studies and tests on real protein datasets which suggest that these modified heuristics are very good at (and are even, in some cases, necessary for) detecting target motifs.

Acknowledgments

For Dr. Tao Jiang, my supervisor, his invaluable supervision in guiding and supporting this work as well as all his support, encouragement and help during my study here, I deeply express my special thanks to him at this opportunity. I would like to express my sincere appreciation to Dr. Todd Wareham for his contribution in the algorithm design and help in reviewing this thesis. Besides, thanks Mr. Chris Trendall for his inputs during algorithm design and implementation and Mr. Peng Zhao and My wife Xin Zhou's cooperation and help during my research period. In addition, I would like to thank Dr. Martin Tompa for his communications with me on [RT98]. Of course, I thank so much Dr. Sanzheng Qiao and Dr. Jeffery Zucker for reviewing my thesis, and Chris Bryce for his technical assistance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Definition of the Problem

A very important application of protein and nucleotide sequence information is the identification of potential targets for drugs. In this identification process, one is interested in finding those regions of the sequence that have important functions within an organism, e.g., regulatory sites in nucleotide sequences or active-site folding domains in protein sequences. Such regions are subsequently used to design assays that screen candidate molecules to see which candidates bind to those regions and hence might be useful drugs.

The ideal way to isolate such functionally-important regions is to know how these sequences function. Such information is very hard to determine.

Fortunately, the nature of sequence mutation provides a relatively quick and useful heuristic for determining such regions. As sequence mutations over the course of evolution are less likely to change parts of the sequence that are important to the function of that sequence, one can isolate functionally-important sequence regions by finding those regions that are conserved over the sequences for a set of species that have descended from a common ancestor (and hence have evolved by mutation from a common ancestral sequence). Such conserved regions are known as **motifs**, and the process of finding them corresponds to the following computational problem:

MOTIF IDENTIFICATION

Input: A set of sequences S over an alphabet Σ , a motif -to-string distance function $d : \Sigma^* \times \Sigma^* \mapsto \mathcal{N}$ [1] and positive integer k and l.

Output: The (possibly empty) collection $M = \{M_1, M_2, \ldots\}$ where each M_i is a set of strings $\{m, r_1, r_2, \ldots\ r_{|S|}\}$ such that m is a string over Σ with minimum length of l and for all i, r_i is a substring of the ith sequence in S and $d(r_i, m) \leq k$.

Here, Σ^* denotes the set of all strings over the alphabet Σ. This problem can also be phased relative to a motif-to-string similarity function if the inequalities are reversed. In this formulation, the string m in each subset M_i

[1] $\mathcal{N}$ is the set of integers

is the motif. There is an extensive literature on deriving motifs relative to various distance functions on strings (see [DEKM98, Gus97] and their references). However, to be truly useful as drug targets, such motifs must often be specific to particular groups of organisms and not occur in certain other groups of organisms. For instance, if one is searching for drugs to kill certain kinds of bacteria in people, one is typically only interested (for financial and legal (if not ethical) reasons) in those drugs that kill the bacteria and not the people. The process of finding the group-specific motifs underlying such useful drugs corresponds to the following modification of the problem above:

TARGET MOTIF IDENTIFICATION

Instance: Sets T and A of strings over an alphabet Σ, a motif-to-string distance function $d : \Sigma^* \times \Sigma^* \mapsto \mathcal{N}$ and positive numbers d_T and d_A.

Solution: A motif m such that for each string $y \in T$, there is a substring y' of y for which $d(m, y') \leq d_T$, and for every substring z' of every sequence z in A, $d(m, z') \geq d_A$ with a lower bound of motif length l.

In this thesis, we propose and implement an algorithm based on the popular Gibbs Sampling motif-finding heuristics [LAB+93, LNL95, NLL95, RT98] that uses avoidance sequences to help detect target motifs. We also present the results of several experiments which suggest that these modified heuristics

are very good at detecting target motifs.

1.2 Terminology

A *motif* is essentially a pattern that occurs in one or more sequence. A motif can be encoded in many ways; for instance, a motif can be encoded as a string, an alignment of strings, or a profile, *i.e.*, a table giving the probabilities of occurrence of all symbols in the sequence-alphabet at each position in the motif (see [Gus97, Chapter 14] and references). To a limited extent, it is possible to transform one type of motif-encoding into another; for instance, a motif encoded as an alignment of strings can be transformed into a profile by computing the frequencies of occurrence of every symbol in each column of the alignment, and this profile can in turn be transformed into a consensus string by selecting for each position in that string the symbol with the maximum frequency of occurrence in that position in the profile. This profile we call it *motif model* (see Table 1.1 as an example). A *motif instance* is a subsequence from a given target sequences that can be considered as a candidate for a motif.

Choosing symbols with the highest probability in each column, we get a string C-ACGA, a *consensus motif*, at the bottom of the Table 1.1. That is: Given a multiple alignment M of a set of strings S, the *consensus character*

Symbol	Motif Position					
	1	2	3	4	5	6
A	10%	0%	95%	0%	5%	70%
C	70%	5%	0%	90%	0%	5%
G	15%	20%	5%	10%	85%	0%
T	5%	5%	0%	0%	5%	5%
-	0%	70%	0%	0%	0%	20%
consensus motif	C	-	A	C	G	A

Table 1.1: An Example Motif Model Extracted From A Multiple Sequence Alignment

of column i of M is the character that minimizes the summed distance to it from all the characters in column i. Let $d(i)$ denote that minimum sum in column i. Since the alphabet is finite, a consensus character for each column of M exists and can be found by enumeration. As one simple special case, if the pairwise scoring scheme scores a match with a zero and a mismatch or a space opposite a character with a one, then the consensus character in column i is the *plurality* character (i.e., the character occurring the most often in column i). Note that the plurality character can be a space. The *consensus motif* S_M derived from alignment M is the concatenation of the consensus characters for each column of M.

Each motif has an associated function that is used to assess how well a given string matches that motif. A motif *matches* a string if the value of that motif's associated distance (similarity) function relative to the motif and

that string is below (above) a specified threshold value. The nature of this function depends on the type of motif-encoding; for instance, if the motif is encoded as a string, the function might be a distance function between pairs of strings, and if the motif is encoded as an alignment of strings, the function might be the value of the optimal alignment of the strings already in the alignment and the new string. When motifs are encoded as strings, two popular matching functions are *Hamming distance*, (which is the number of positions at which symbols differ in two strings of equal length) and *edit distance* (which is essentially the minimum number of symbol substitutions, insertions, and deletions that must be applied to transform one string into another). Note that both of these functions can be rephrased as similarity functions, which measure the number of identical-symbol positions and the maximum number of identical-symbol positions relative to a padding of both sequences with gap symbols, respectively. A motif that matches one or more substrings of a given sequence is said to *appear* in that sequence.

A motif can allow gaps, which correspond to positions at which insertions or deletions can occur in the matching of that motif to a given string. These gaps can be explicit in the motif itself (either as special symbols in a motif-string or a profile or as the gaps in an alignment of strings) or implicit in that motif's associated matching function. For instance, the latter would be

the case if a motif is encoded as a string and the matching function is edit distance. If the motif incorporates gaps, it is called a *gapped motif*; else, it is an *ungapped motif*.

In the example of Figure 1.1, the first pattern (e.g. AAAGCTAG) corresponds to an gapped non-target motif and the second pattern (e.g. CCGAGAAT) corresponds to an gapped target motif.

In the target identification problem, denote the sets T and A as the *target sequences* and *avoidance sequences*, respectively. Any motif that appears in both the target and avoidance sequences is a *non-target motif* and motif that appears only in the target sequences is a *target motif*.

1.3 Applications of Target Motif Identification

Target motifs correspond to patterns that characterize sequences in a subset of a given set of sequences, and hence distinguish sequences in that subset from all other sequences in the larger set. This ability makes target motifs valuable in several pharmaceutical applications. For instance, target motifs can be used in the design of diagnostics for the presence or absence of some subset of a group of bacteria, *e.g.*, diagnostics for the presence of pathogenic strains of *E. Coli* in an environment known to contain multiple strains of *E.*

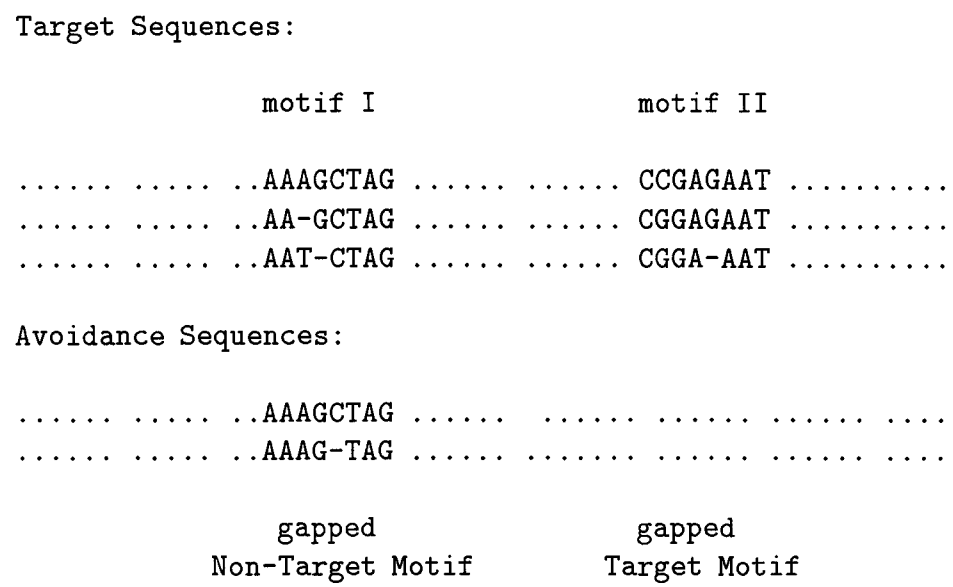

Figure 1.1: An example of gapped target motif. Motif I is a non-target motif since it presents in the avoidance sequences; motif II is a target motif because it does not show similar pattern in the avoidance sequences. Both motif I and motif II are gapped motif since they contain gaps

coli. Target motifs may also be useful as candidate sequence targets in the design of broad spectrum drugs that act on a specified group of organisms without affecting another specified group of organisms, *e.g.*, antibiotics that disrupt particular genes or gene products in pathogenic strains of *E. coli* while leaving the corresponding genes and gene products in human beings and normal human-internal microbial flora intact. Such sequence targets may be used either directly as targets for anti-sense therapeutics or indirectly as components of assay for the screening of other potential compounds against group-specific sequences that are conserved in a group of interest and hence may be crucial to the functioning of organisms in this group. These applications and others are discussed in more detail in [LL+99].

It is very important to stress that many of the target motifs produced by the target motif finding algorithms may not be useful as diagnostics or drug targets for various reasons, *i.e.*, motif occurring in sequences other than those considered in an organism, secondary-structure interference with binding. However, they should prove a first approximation to sets of useful targets and diagnostics, and thus make the diagnostic and drug development process faster and cheaper by narrowing the range of sequences or compounds that must be considered in subsequent laboratory-intensive rounds of testing.

1.4 Previous Results for Target Motif Identification

Ito *et al.* [ISNH94] gave a low-order polynomial-time algorithm for a special case of the target motif identification problem in which the motif is a string, the matching function is edit distance, and the target motif is required to be a substring of each of the target sequences. Lanctot *et al.* [LL+99] considered the case in which the motif is a string, the matching function is Hamming distance, and all target sequences are of the same length as the motif. They showed that this problem (which they called *Distinguishing string selection*) is NP-hard and gave a polynomial-time approximation algorithm that is guaranteed to produce a motif whose distance to the target sequences is within a factor of 2 of optimal. Unfortunately, the high order of the polynomial in this algorithm's time complexity renders it useless in practice.

1.5 Summary of My Results

The first contribution of my thesis is a set of simple modifications to the Gibbs Sampling heuristic algorithms for finding ungapped [LAB+93] and gapped

[RT98] motifs which allow these heuristics to find target motifs. These modified algorithms actually solve a relaxed version of the target motif identification problem in which a derived target motif is only guaranteed to be far relative to some threshold from the given avoidance sequences under a given similarity function and is assumed to be close to the target sequences under the probabilistic distance function implicit in the Gibbs motif-finding heuristic. However, both algorithms operate in low-order polynomial time and the latter of these algorithms is the only known algorithm that can derive gapped target motifs.

The second contribution of my thesis is a set of experiments on simulated and real datasets whose results establish the following two points: 1) that algorithms such as the one proposed in this thesis which integrate avoidance information into target motif search are useful because the naive method for solving the target motif identification problem may fail under certain conditions (namely, when weakly-conserved target motifs exist in the same sequence with strongly-conserved non-target motifs); and 2) the algorithm described in this thesis finds the target motifs in given sequence datasets quickly under a variety of conditions.

This thesis is organized as follows: Chapter 2 describes the algorithms, which consists of a description of the basic Gibbs Sampling algorithm for

finding ungapped target motifs [LAB+93] and a generalization of this algorithm [RT98] to handle gapped motifs. Chapter 3 describes how the implementation is organized and its main data structures as well as performance analysis. Chapter 4 reports the simulation studies. Chapter 5 reports some experiments on real sequence datasets. Chapter 6 discusses about performance of our algorithm on real datasets. Chapter 7 is the conclusions and future work.

Please note that some of the aspects and conclusions given in this thesis appeared previously in RECOMB poster in 1999 [JTWZ99] and the manuscript to be submitted to Journal of Computation Biology [JTW99].

Chapter 2

Algorithms

2.1 Basic Ideas of Gibbs Sampling Algorithm

Gibbs Sampling can be viewed as an instance of a general stochastic strategy for determining the parameters of a statistical model relative to a given data set. This strategy starts with some setting of parameter-values and iteratively changes the value of one parameter at a time by assuming that the remaining parameters are correct and invoking Bayes' Theorem until all parameters converge to stable (if not optimal) values (see [LAB+93, LNL95] and references for details). With reference to the motif identification problem, the model is a motif encoded as an alignment of strings, the parameters are the positions of the motif within each sequence in a given set S (the

motif-instances), and the stochastic heuristic modifies these motif-instances one sequence at a time, one sequence per iteration, until the alignment of these motif-instances de notes a stable (if not optimal) motif.

The general algorithm for Gibbs Sampling motif identification is given in Table 2.2. Though several steps of this algorithm may be stochastic, the primary stochastic element is the stochastic selection performed in Step 8. Under stochastic selection, an element in a set is selected at random relative to the probabilities derived by normalizing the weights assigned to the elements in that set. This type of selection is intuitively more appealing than a deterministic selection that would always select the highest- or lowest-weighted value because stochastic selection can allow a local-search heuristic algorithm to escape from (and hence avoid being trapped in) local optima.

The Gibbs motif identification algorithms described in the remainder of this thesis all follow the outline of the algorithm given in Table 2.2. These algorithms implement this basic algorithm in different ways, but can ultimately be characterized by how they answer the following two questions:

1. What constitutes a motif model?

2. How is such a motif model scored against the motif-instances in a selected sequence?

program Stochastic Motif Identification

Input: A set S of sequences over some alphabet.
Output: A candidate motif for S.

boolean *finished*

begin
1. Select initial motif-instances in the sequences of S.
2. Create initial motif-model from motif-instances.
3. *finished* := **false**
4. **while not** *finished* **do**
5. Select sequence s from S.
6. Construct motif-model from motif-instances in $S - \{s\}$.
7. Weight all possible motif-instances in s relative to thee motif-model derived above.
8. Stochastically select a new motif-instance x for s relative to these weights.
9. Update motif-instance information for s relative to x.
10. Check if motif-model has converged and process is finished.
11. Output motif-instances for S.

end

Table 2.2: Basic Algorithm for Stochastic Motif Identification.

2.2 Ungapped Motif Identification

2.2.1 General Strategy

The first algorithm for identifying motifs by Gibbs sampling was given in [LAB+93]. This algorithm finds ungapped motifs of a pre-specified length W. In this algorithm, a motif is modeled as a collection of $W+1$ multinomial probability distributions over the sequence alphabet Σ, where the first W of

these distributions correspond to a profile-encoding of the motif, *i.e.*, the first W of these distributions correspond to the frequencies of occurrence $q_{i,j}$, $1 \leq i \leq W$ and $1 \leq j \leq |\Sigma|$, of symbol j at position i in the motif, and the final distribution corresponds to the "background" frequencies of occurrence p_j, $1 \leq j \leq |\Sigma|$, of symbol j in parts of the sequences that are not in the motif. A candidate motif-instance string $x = x_1 x_2 \cdots x_W$ over alphabet Σ is evaluated against the motif model in terms of the ratio [1]

$$Q_x / P_x = \prod_{i=1}^{W} (q_{i,\mathbf{sym\_ind}(x_i)} / p_{\mathbf{sym\_ind}(x_i)})$$

where $\mathbf{sym\_ind}(s)$ is the index of symbol s in Σ. For numerical reasons, this product of ratios is more often computed as an equivalent sum of log-ratios known as the *F-value*,

$$F = \sum_{i=1}^{W} \sum_{j=1}^{|\Sigma|} c_{i,j} \log q_{i,j} / p_j$$

where $c_{i,j}$ is the unnormalized count of the number of occurrences of symbol i at position j in the motif, and $c_{i,j}$, $q_{i,j}$, and p_j are computed from the motif model based on all motif-instances in $S - \{s\}$ and the candidate string x.

Intuitively, by seeking motif models that maximize the ratio Q_x / P_x,

[1] Q_x are the probabilities of generating each segment x according to the current motif model; P_x are the probabilities of generating these segments by background.

the algorithm is searching for the motif model whose collective symbol-occurrence distribution is probabilistically the most distinct from the background symbol-occurrence distribution. As such, the distance function encoded in this algorithm is a variant of the *Kullback-Leibler distance*

$$H(Q||P) = \sum_x Q(x) \log(Q(x)/P(x))$$

which gives a measure of the distinctness of probability-distributions Q and P ($H(Q||P)$ is also known as the *relative entropy* of Q to P). This connection is more easily seen in the re-formulation of the ratio Q_x/P_x in terms of F.

2.2.2 Algorithm

The algorithm in [LAB+93] is described in Tables 2.3, 2.4 and 2.5. Several matters are worth noting about this algorithm. First, due to various problems associated with storing and operating on very small real numbers accurately in a computer, For numerical reasons to avoid underflow, the probability ratio Q_x/P_x will typically be computed as the the log-odds ratio [DEKM98, Section 3.6]:

$$\sum_{i=1}^{W} (\log q_{i,\mathbf{sym\_ind}(x_i)}) - (\log p_{\mathbf{sym\_ind}(\mathbf{x_i})}))$$

Second, the pseudocounts in procedure Construct_Motif_Model are designed to handle the problems that avoids zero probabilities and "normalizes" the frequency-occurrence counts obtained from the limited data set in S to more

accurately reflect the underlying statistical distributions in the model. In this method, probability estimates are obtained in a two-step process. First, pseudocounts $MM.psu[i]$ for each possible symbol i in the alphabet (amino acids, in the case of proteins) added to the observed counts are divided by the total counts over all symbols (observed plus pseudocounts), to obtain the probability of each symbol. That is, the expected probability of a letter i at motif model position j is:

$$q_i = \frac{cm[i][j] + psu[i]}{|n| + |z|}$$

where $|n|$ is the number of target sequences and $|z|$ is the summation of pseudocounts. It is claimed that the algorithm if each $psu[i]$ is multiplied by a square root of N, where N is the number of sequences, works better [LAB+93, p. 214]. The statistical problems associated with inferring motif models from a small number of sequences often requires more complex and computationally-expensive pseudocounts [DEKM98, Section 5.6]. Finally, the termination of the algorithm is judged in terms of a quantity *F-score* and a integer value k, i.e. it is terminated if the F-value has not been increased in number of k iterations.

```
program  Ungapped Gibbs Motif Identification

Input:   S, a set of sequences over some alphabet with num_sym symbols,
         W, an expected motif-width.
         NI, convergence parameter
Output:  A set of length-W motif-instances for S.

record  S          /* Input sequences */
  integer  N                    /* Number of sequences */
  integer  SLEN[N]                /* Length of each sequences */
  char  *SEQ[N]                 /* Sequences */

record  MM   /* Motif model */
  integer  W                    /* Width of motif */
  integer  mi_start[1...N]      /* Start-points of motif-instances */
  integer  cm[1...W][1...num_sym]/* Motif position symbol-counts */
  integer  cb[1...num_sym]      /* Background symbol-counts */
  integer  tot_cb
  real  psu[1...num_sym]        /* Symbol pseudo-counts */
  real  tot_psu
  real  q[1...W][1...num_sym]/*Motif position symbol probabilities*/
  real  p[1...num_sym]          /* Background symbol probabilities */

integer  s
boolean  finished

begin
  Initialize_Motif_Model(S, W, MM)
  finished := false
  while  not  finished do
     Randomly select sequence s from S.
     Construct_Motif_Model(S, s, MM)
     Select_New_Motif_Instance(S, s, MM)
     finished := Check_If_Finished(S, MM, NI)
  Output motif-instances for S.
end
```

Table 2.3: Ungapped Gibbs Motif Identification: Data Structures and Main Program.

```
procedure Initialize_Motif_Model(S, W, MM)
begin
  Initialize MM by setting all value to zeroes
  Randomly select motif start-position for each sequence in S
  Build an initial motif model based on the initial motif-instances
end     /* Of INITIALIZE_MOTIF_MODEL */

procedure Construct_Motif_Model(S, s, MM)
begin
   Compute MM.cm[i][j], 1 ≤ i ≤ MM.W and 1 ≤ j ≤ num_sym,
     to the number of occurrences of symbol j in column i of the
     implicit motif-instance alignment
       (excluding the row for the motif-instance in sequence s).
   Compute MM.cb[j], 1 ≤ j ≤ num_sym, to the number of background
    occurrences of symbol j, i.e., the number of occurrences of symbol j
    in all sequences in S except s that are not in any motif-instance
   Set MM.tot_cb = Σ_{j=1}^{num_sym} MM.cb[j].
   Compute_Q_{i,j}(S, MM)
   Compute_P_j(S, MM)
end     /* Of CONSTRUCT_MOTIF_MODEL */

procedure Compute_Pseudocount(S, MM)
begin
   Compute MM.psu[j], 1 ≤ j ≤ num_sym,
         to the pseudocount for each symbol and set
   Set total pseudocount MM.tot_psu = Σ_{j=1}^{num_sym} MM.psu[j].
end     /* OF Compute_Pseudocount */

procedure Output_Motif(S, MM)
begin
   for i = 0 to S.N do
     for j = MM.mi_start[i] to MM.W do
        output S.SEQ[i][j]
end     /* OF Output_Motif(S, MM) */
```

Table 2.4: Ungapped Gibbs Motif Identification: Functions and Procedures (Cont'd).

function **sym_ind**(s)
Returns index of symbol s in alphabet.

procedure Compute_$Q_{i,j}$(S, MM)
begin
for $i = 1$ **to** W **do**
for $j = 1$ **to** num_sym **do**
$MM.q[i][j] = \frac{MM.cm[i][j]+MM.psu[j]}{(MM.N-1)+MM.tot\_psu}$
end /* OF Compute_Pseudocount */

procedure Compute_P_j(S, MM)
begin
for $j = 1$ **to** num_sym **do**
$MM.p[j] = \frac{MM.cb[j]+MM.psu[j]}{MM.tot\_cb+MM.tot\_psu}$
end /* OF Compute_Pseudocount */

procedure Select_New_Motif_Instance(S, s, MM)
begin
for each substring $x = x_1x_2 \cdots x_W$ of length W in sequence s **do**
Compute $A[i] = Q_x/P_x$, where i is the start-point of x in s
$Q_x = \prod_{i=1}^{W} MM.q[i][\textbf{sym\_ind}(x_i)]$,
$P_x = \prod_{i=1}^{W} MM.p[\textbf{sym\_ind}(x_i)]$.
Select new motif-instance x' for s by stochastically sampling
over the normalized weights
$A'[i] = A[i]/\Sigma_{j=1}^{S.len[s]} A[j]$.
Update motif-instance information for s relative to x'.
end /* Of SELECT_NEW_MOTIF_INSTANCE */

function Check_If_Finished(S, MM, NI)
begin
Recompute $MM.q[i][j]$ and $MM.p[j]$
$F = \sum_{i=i}^{W} \sum_{j=1}^{num\_sym} MM.c[i][j] \log \frac{MM.q[i][j]}{MM.p[j]}$.
if F has not been improved more than NI iterations
return **true**
else **return** **false**
end /* Of CHECK_IF_FINISHED */

Table 2.5: Ungapped Gibbs Motif Identification: Functions and Procedures (Cont'd).

2.2.3 Time Complexisity

At present, outside of some rules-of-thumb derived from practical experience [LAB+93, p. 213], it is not known how many iterations will be required for the algorithm to converge on a motif model for a given data set. Hence, the actual time complexity of this algorithm cannot be given. However, it is possible to give the time complexity of each iteration. The time and space complexities of each of the functions and procedures used in the algorithm are as follows:

- **Initialize_Motif_Model**: $O(N)$ time, $O(1)$ space.
- **Construct_Motif_Model**: $O(NL + W|\Sigma|)$ time, $O(1)$ space.
- **Select_New_Motif_Instance**: $O(LW)$ time, $O(L)$ space.
- **Check_If_Finished_Instance**: $O(NL + W|\Sigma|)$ time, $O(1)$ space.

The sequence and motif-model data structures require $O(NL+W|\Sigma|)$ space. Hence, if I is the number of iterations the algorithm requires to converge, the time and space complexities of the algorithm as a whole are $O(I(NL+W|\Sigma|))$ and $O(NL + W|\Sigma|)$, respectively.

Many modifications of this basic algorithm are possible. For instance, it is possible to estimate the length W of the best motif model over several runs

of the algorithm described above using a parameter G [LAB+93], which is related to F-value; more computationally and statistically complex methods, e.g., column sampling/fragmentation. are described (albeit not in algorithmic terms) in [LL95, LNL95, NLL95]. Besides, the algorithm can be easily adapted to allow multiple motif models at one time; however, this will also increase the number of iterations require to converge [LAB+93, page 210]. Various other modifications dealing with preserving consistent orders across multiple models are described in [LAB+93, LL95, LNL95, NLL95].

The algorithm described above is the most basic version of the Gibbs sampling motif-finding algorithm, in that it assumes that there is one motif in the given set of sequences, one copy of that motif in each sequence, and the length of that motif is known. Modifications of this basic algorithm that allow it to automatically set motif length and automatically determine both the number of copies of a motif in each sequence as well as the number of motifs present in the set of given sequences are described (albeit often in statistical rather than algorithmic terms) in [LAB+93, LNL95, LL95, NLL95].

2.2.4 Identifying Gapped Motifs

2.2.5 General Strategy

Though a strong case can be made for the utility of ungapped motifs on the basis of the domain-structure of proteins [LAB+93], there are also occasions when gapped motifs are desirable, e.g., when non-Hamming distance measures are used in motif identification. Two algorithms for identifying gapped motifs by Gibbs sampling have been given in the literature. The first algorithm, which incorporates constraints that favor collinearity and close spacing of multiple motifs into the algorithm given in [LAB+93], is mentioned briefly on page 210 of [LAB+93] and is partially sketched in [LL95]; however, this algorithm did not significantly increase performance in practice relative to the original algorithms and was more susceptible to being trapped in local optima [LAB+93, p. 210]. The second algorithm (given in [RT98]) was originally designed for finding multiple occurrences of a gapped motif in a single sequence; however, it can be trivially reformulated to look for a single occurrence of a motif in each sequence of a given set of sequences by making each motif-instance occur in a distinct sequence. This algorithm will be described in more detail below.

The algorithm in [RT98] encodes a motif as an alignment of strings, each of which is a motif-instance substring from a different sequence. A candidate

motif-instance string x is evaluated against the motif in terms of the score of the best pairwise alignment of x and the motif-model alignment. In this pairwise alignment, each symbol r of x is aligned with a column j of n symbols drawn from the motif-model alignment. The column j' of $n+1$ symbols composed of r and j has the score

$$\sigma(r,j) = (\sum_{k:P_k^{j'} \neq 0} P_k^{j'} \log_2(P_k^{j'}/B_k)) + (g \cdot p_g)$$

where $P_k^{j'}$ is equal to $(C_k^{j'} + gB_k)/(n+1)$, $C_k^{j'}$ is the number of occurrences of symbol k in j', B_k is the number of occurrences of k in the background portions of the sequences that are not part of any motif-instance, g is the number of gaps in j', and p_g is a user-defined penalty associated with creating a gap. As defined above, the score $\sigma(r,j)$ is always non-negative, and hence could encourage the proliferation of poorly-conserved columns in the optimal alignment; hence, the authors recommend that the expected value of $\sigma(r,j)$ be subtracted from each column's score to reduce the expected score to zero. By analogy with the algorithm in [LAB+93], we will call the score of an alignment relative to $\sigma(r,j)$ the F-value of that alignment. Initially, all of the substrings comprising the motif alignment are of a pre-specified length W. However, over the course of executing the algorithm, the length of the alignment will change subject to the constraints on gap and column insertion exercised by the value of p_g.

In practice, rather than evaluating the pairwise alignment of the motif-model alignment and each substring x of sequence s in Step 7 of the algorithm given in Table 2.2, the motif-model alignment is aligned with all of s simultaneously using a variant of end-space-free alignment [Gus97, Section 11.6.4] such that the score associated with the best alignment of the motif alignment and the substring of s ending at position i is given in position (w, i) of the dynamic programming matrix. This not only lowers the computation time associated with Steps 7 and 8 from $O(W^2L)$ to $O(WL)$, where W is the length of the motif-model alignment and L is the length of the longest given sequence, but also allows the length of the motif alignment to change as symbols within individual sequences are inserted and deleted (see [RT98] for details).

The distance function encoded in this algorithm is a variant of the Kullback-Leibler distance, and hence has the same justification as that given for the algorithm in [LAB+93]. Note, however, that the admittedly *ad hoc* manner in which gap-penalties are introduced into the distance function renders this connection more tenuous.

2.2.6 Algorithm

The gapped Gibbs motif identification algorithm is given in Tables 2.6, 2.7, 2.8, 2.9 and 2.10.

To quickly pick up a sampled candidate motif instance, we stored the weights to an one dimensional array A (at the bottom of Table 2.9) and do a binary search to find the value. This takes $O(\log L)$ time, where L is the length of the longest sequence.

2.2.7 Time Complexity

As is the case for the algorithm in [LAB+93], there is no known bound on the number of iterations. However, it is still possible to give the time complexity of each iteration. The time and space complexities of each of the functions and procedures used in the algorithm are as follows. In the expressions below, let W' be the length of the alignment of the motif-instances. Though W' is upper-bounded by NL in the worst case, i.e., the alignment consists of all strings such that none overlap in the alignment, it will in practice be $O(W)$.

- **Initialize_Motif_Model**: $O(NW)$ time, $O(1)$ space.
- **Construct_Motif_Model**: $O(NL + W|\Sigma|)$ time, $O(1)$ space.
- **Compute_Cost_Matrix**: $O(|\Sigma|W')$ time, $O(1)$ space.

```
program  Gibbs Sampling Algorithm on Gapped Motif Identification

Input:   S, a set of sequences over num_sym symbols,
         W a motif-width W. NI, a convergence value, G, gap penalty

Output:  An alignment of motif-instances for S.

record  S   /* Input sequences */
  integer  N                          /* Number of sequences */
  integer  len[1...N]                 /* Lengths of sequences */
  char     sym[1...N][1...len[i]]     /* Symbols in sequences */

record  MM   /* Motif model */
  integer  N                          /* Number of sequences */
  integer  W                          /* Width of motif */
  integer  mi_start[1...N]            /* Start-points of motif-instances */
  integer  mi_finish[1...N]           /* End-points of motif-instances */
  integer  alen                       /* Length of motif */
  char     asym[1...N][1...alen]      /* Motif-instance alignment */
  integer  cm[1...alen][1...num_sym]
                  /* Motif-instance alignment position symbol-counts */
  integer  cg[1...alen]   /* gap-counts */
  integer  cb[1...num_sym]            /* Background symbol-counts */
  integer  tot_cb
  real     q[1...W][1...num_sym] /* symbol probabilities */
  real     p[1...num_sym]             /* Background symbol probabilities */

integer  s
boolean  finished

begin
   Gibbs Sampling Main Loop
end
```

Table 2.6: Gapped Gibbs Motif Identification: Data Structures and Main Program.

```
begin
  Initialize_Motif_Model(S, W, MM)
  finished := false
  while not finished do
    Randomly select sequence s from S.
    Construct_Motif_Model(S, s, MM)
    Select_New_Motif_Instance(S, s, MM)
    finished := Check_If_Finished(S, MM, NI)
  Output motif-instance alignment for S.
end
procedure Initialize_Motif_Model(S, W, MM)
begin
  Initialize all values to zeros
  Randomly select the motif start-position for each sequence in S.
  Initialize the motif-instance alignment to the ungapped
   alignment of the selected motif-instances.
end      /* Of INITIALIZE_MOTIF_MODEL */

procedure Construct_Motif_Model(S, s, MM)
begin
  Set MM.cm[i][j], 1 ≤ i ≤ MM.W and 1 ≤ j ≤ num_sym,
   to the number of occurrences
   of symbol j in column i of the implicit motif-instance
    alignment (excluding the row for the motif-instance in sequence s).
  Set MM.cg[i], 1 ≤ i ≤ MM.alen, to the number gaps in column
   i of the motif-instance
   alignment (excluding the row for the motif-instance in sequence s).
  Set MM.cb[j], 1 ≤ j ≤ num_sym, to the number of background
    occurrences of symbol j, i.e.,
   the number of occurrences of symbol j in all sequences in S
    except s that are not in any
   motif-instance, and set MM.tot_cb = Σ_{j=1}^{num_sym} MM.cb[j].
  for j = 1 to num_sym do
    MM.p[j] = MM.cb[j]/MM.tot_cb
  for i = 1 to W do
    for j = 1 to num_sym do
      MM.q[i][j] = (MM.cm[i][j] + (MM.cg[j] * MM.p[j]))/MM.N
end      /* Of CONSTRUCT_MOTIF_MODEL */
```

Table 2.7: Gapped Gibbs Motif Identification: Data Structures and Main Program (Cont'd).

```
FUNCTION sym_ind(s)
  Returns index of symbol s in alphabet.

procedure  Select_New_Motif_Instance(S, s, MM)
real  C[1...num_sym + 1][1...MM.alen + 1]
real  DPM[0...S.len[s]][0...MM.alen]
real  A[1...S.len[s]]
begin
  Compute_Cost_Matrix(MM, C)
  Compute_DP_Matrix(S, s, C, DPM)
  Weigh_Normalization(S, s, C, DPM)
  Select the end-point of the new motif-instance x' for s by
     stochastically sampling over the normalized weights by binary search
  For selected end-point i, traceback in DP matrix from DPM[i][MM.alen]
    to column 1 of the DP matrix to determine start-point x'
  Update motif-instance information for s relative to x'.
end      /* Of SELECT_NEW_MOTIF_INSTANCE */

procedure  Compute_DP_Matrix(S, s, C, DPM)
begin
  for  i = 0 to  S.len[s] do
     DPM[i][0] = 0.0
  for  j = 0 to  MM.alen do
     DPM[0][j] = DPM[0][j - 1] + C[num_sym + 1][j - 1]
  for  i = 1 to  S.len[s] do
     for  j = 1 to  MM.alen do
        DPM[i][j] = max(
           DPM[i - 1][j - 1] + C[sym_ind(S.sym[s][i])][j],
         % Alignment-column /
   %     symbol match
           DPM[i - 1][j] + C[sym_ind(S.sym[s][i])][MM.alen + 1],
 % Symbol gap
           DPM[i][j - 1] + C[num_sym + 1][j]
                    % Alignment-column gap
         )
end      /* Of COMPUTE_DP_MATRIX */
```

Table 2.8: Gapped Gibbs Motif Identification: Functions and Procedures (Cont'd).

procedure Compute_Cost_Matrix(MM, C)
begin
/* Initialize all entries in cost matrix C to 0.
for $i = 1$ **to** num_sym **do**
add the num_sym count
for $j = 1$ **to** $MM.alen$ **do**
compute $C[i][j]$by using the score function $\sigma(r, j)$

/* Initialize C for entries in which symbol is a gap. */
for $j = 1$ **to** $MM.alen$ **do**
add a gap count to the column
for $k = 1$ **to** num_sym **do**
compute C[num_sym+1][j] by using the score function $\sigma(r, j)$

/* Initialize C for entries in which alignment-column is all gaps.
for $i = 1$ **to** num_sym **do**
initialize MM.cm[MM.alen+1] = 1.0
for $k = 1$ **to** num_sym **do**
$C[i][MM.alen + 1] = 1.0/N(MM.cm[MM.alen + 1][k] + MM.p[k]t)$
$\log(\frac{(1/N(MM.cm[MM.alen+1][k]+MM.p[k]t))}{MM.p[k]})$

/* Incorporating gap penalty
for i=1 **to** MM.alen **do**
for j=1 **to** num_sym **do**
$C[j][i] = C[j][i] - (\Sigma C[j][i] + numberofgapsinthecolumn * penalty)$
$C[num\_sym][i] = C[num\_sym][i] - (\Sigma C[j][i] + numofgaps$
$inthecolumn * penalty)$
$C[i][MM.alen] = C[i][MM.alen] - t * columngappenalty$

end /* Of COMPUTE_COST_MATRIX */

procedure Weight_Normalization(S, s, C, DPM)
begin $A[i] = DPM[i][MM.alen]/\Sigma_{j=1}^{S.len[s]} DPM[j][MM.alen]$,
$1 \leq i \leq S.len[s]$.
end

Table 2.9: Gapped Gibbs Motif Identification: Functions and Procedures (Cont'd).

```
function  Check_If_Finished(S, MM, NI)
begin
   Recompute MM.q[i][j] and MM.p[i][j] as in Construct_Motif_Model
   relative to all sequences in S. But add the new motif instance
   Compute F-score of motif model as :
        sum_{i=i}^{W} sum_{j=1}^{num_sym} MM.c[i][j] log MM.q[i][j] / MM.p[j].
  if  F-score of motif model has been improved
       counter = 0;
       Update output motif model;
  else  counter++;
  if  counter ≤ NI iterations then
     return  true
  else
     return  false
end      /* Of CHECK_IF_FINISHED */
```

Table 2.10: Gapped Gibbs Motif Identification: Functions and Procedures (Cont'd).

- **Compute_DP_Matrix**: $O(LW')$ time, $O(1)$ space.

- **Select_New_Motif_Instance**: $O(|\Sigma|W'+LW'+(L+W')N) = O(LW'|\Sigma|N)$ time, $O(|\Sigma|W' + LW') = O((|\Sigma + L)W')$ space.

- **Check_If_Finished_Instance**: $O(NL + W|\Sigma|)$ time, $O(1)$ space.

The sequence and motif-model data structures require $O(NL + NW' + W'|\Sigma|) = O(NL+(N+|\Sigma)W')$ space. Hence, if I is the number of iterations that the algorithm requires to converge, the time and space complexities of the algorithm as a whole are $O(I(NLW'|\Sigma|))$ and $O(NL+(N+L+|\Sigma|)W')$, respectively.

The algorithm above can be seen as a generalization of [LAB+93] that operates purely in terms of optimizing the F-value rather than the log-odds ratio; hence, if appropriate care is taken in constructing and operating on the dynamic programming matrix, this algorithm can construct gapped or ungapped motifs. This algorithm can also be modified to handle multiple motif models along the lines described in the previous sections. One very nice advantage of this algorithm is that it does not need to invoke complex statistical or computational machinery to decide on the optimal width W for a motif model – rather, the width W evolves as the length of the motif-instance alignment over the execution of the algorithm to best fit the given data.

2.3 Identifying Target Motif

There are two obvious points in the algorithm given in Table 2.2 (and hence in the algorithms described in Section 2.2.2 and 2.2.6) at which avoidance sequence information can be used to influence target motif search – namely, Step 1 (when the initial motif-instances are selected) and Step 7 (when the weights of the potential motif-instances for a given sequence are computed). To implement this influence, we need an easily-computable measure of how

much a candidate motif-instance is like the substrings of the avoidance sequences (and hence how fervently this motif-instance must be avoided). We will first discuss the computation of this measure and then sketch how it can be applied in Steps 1 and 7 of the algorithm.

2.3.1 Construction of Avoidance Sequence Model

The most obvious measure is the score of the best alignment of the motif-instance against every substring in the avoidance sequences under an appropriate distance function. Although this approach is rigorous and appropriately values approximate matches, it can be computationally prohibitive if there are a number of avoidance sequences or these sequences are long. An alternative approach is to describe substrings of avoidance sequences in terms of the parameters of a statistical model. As the number of parameters is typically much smaller than the amount of data in the sequences, there is a loss of information; however, this loss is counterbalanced by a corresponding increase in the computational efficiency of evaluating motif-instances against a compact model.

In this thesis, we use the first-order correlation model θ_A, which encodes the probabilities $P(a|b)$ of symbol a occurring in the avoidance sequences, given that symbol b occurred immediately before symbol a; in practice, $P(a|b)$

is approximated by the frequency of occurrence of substring ba in the avoidance sequences. For a string $x = x_1 \ldots x_n$, the likelihood of the string x being produced by the model θ_A (and hence the weight of an instance of an ungapped motif relative to this model) is

$$P(x|\theta_A) = \prod_{i=1}^{n-1} P(x_{i+1}|x_i)$$

To evaluate such a model against an instance of a gapped motif, treat each insertion as matching all symbols, ignore all deletions, and multiply the appropriate probabilities as before.

2.3.2 Use of Avoidance Sequence Model

Consider now how this avoidance correlation model is used to modify Steps 1 and 7 of the general algorithm. In the case of Step 1, the potential motif-instances are weighted relative to the avoidance correlation model and a motif instance can be selected either deterministically or stochastically relative to these weights. In the case of Step 7, the weight w_x associated with a possible motif-instance x is in turn weighted by the odds that this motif instance doesn't appear in the avoidance sequences, *i.e.*:

$$w_x \cdot \frac{1 - P(x|\theta_A)}{P(x|\theta_A)}$$

(this assumes that the events of the motif model being close to the target sequences but distant from the avoidance sequences are independent, which

seems to be a reasonable assumption). Thus the log odds ratio that evaluates the similarity of a particular subsequence to the pattern while being dissimilar to a model of the avoidance sequences is

$$\log w_x + \log \frac{1 - P(x|\theta_A)}{P(x|\theta_A)}$$

2.3.3 Postprocessing for Candidate Motifs

The final modification proposed here is to embed the algorithm as modified above in a postprocessing loop that terminates only when either the number of iterations of the postprocessing loop exceeds a user-defined bound B_R or the produced motif has similarity greater than some threshold T_P to any substring in the avoidance sequences. This threshold is checked by computing the consensus string associated with the motif, computing an appropriate similarity measure (Hamming similarity in the case of ungapped motifs and edit similarity in the case of gapped motifs) between this consensus string and each substring of the avoidance sequences, and then determining if the maximum of these scores is greater than T_P. Note that this consensus string is computed as described in the *Terminology* section in the Introduction; in the case of gapped motifs, gaps are counted like any other symbol, and if the gaps have the maximum number of occurrences in a column, that column has no associated symbol in the consensus string. This postprocessing loop

is necessary because similarity-thresholds are not explicitly evaluated in the algorithms considered here and, on the whole, it is more important that a produced motif be the required distance from all substrings of the avoidance sequences. For this reason, all motifs produced by these modified algorithms are actually *candidate target motifs*, and will be referred to as such in the remainder of the thesis.

The time complexity for postprocessing is M, where M is the time complexity of the method used. If we use Hamming distance measure the time complexity is $O(L)$, where L is the total length of avoidance sequences. If we use Edit Distance measure, the time complexity is $O(L * K)$, where K is the motif length.

2.4 Some Issues Concerning the Improvement of Motif Detection

2.4.1 Phaseshift

One defect of Gibbs sampling algorithm is the "phase" problem [LAB+93]. The strongest motif may begin, for example, at positions 7, 19, 8, 23, and so forth within the various sequences. However, if the algorithm happens to choose $a_1 = 9$ and $a_2 = 21$ in an early iteration, it will then most likely

proceed to choose $a_3 = 10$ and $a_4 = 25$. In other words, the algorithm can get locked into a non-optimal "local maximum" that is a shifted form of optimal pattern. To reduce this problem, we can apply the *phaseshift* technique [LAB+93]. That is to insert another step into the the Gibbs sampling process. E.g. after every M iterations, we compare the current set of a_k with sets shifted left and right by up to a certain number of letters. Probability ratios may be calculated for all instances and a random selection is made among them with appropriate corresponding weights.

2.4.2 Ranking Multiple Target Motifs

It is practical that algorithm may output multiple target motifs. These multiple target motifs can provide more options for drug designers. The distinct models can be ranked according to two parameters. One is the F-value; the other is the separation distance. The *separation distance* $S = \min_i d(T_i, c) - \max_i d(A_i, c)$, where c is the consensus string associated with the candidate target motif, $d(x, y)$ returns the maximum similarity-score of y to any substring of x under an appropriate similarity measure (Hamming similarity in the case of ungapped motifs and edit similarity in the case of gapped motifs), and $T_i(A_i)$ is the ith target (avoidance) sequence. The separation distance essentially measures the minimum number of symbols that

need to change before the worst of the best matches of the target motif in the target sequences is the same as the best match of the target motif in the avoidance sequences. In certain applications, this is also a direct measure of the utility of a motif – for instance, in the case of DNA sequences, the separation distance is the minimum number of bases in a consensus string associated with a candidate target motif that must mispair before the complement of that sequence can form a duplex and hence interact with some substring of an avoidance sequence.

Chapter 3

Implementation

Based on the algorithm we discussed at previous chapter. a software package was developed. We call it TMIT (pronounced "team made"), which is short for Target Motif Identification Tool. The package includes three parts (see Figure 3.1):

1. *The DNA and protein sequence simulator*: it generates three kinds of datasets to meet the need of simulation test #1, # 2 and # 3.

2. *The main algorithm*: the ungapped Gibbs sampling algorithm and gapped Gibbs sampling algorithms.

3. *The motif evaluation processing*: two testing algorithms to satisfy the evaluation processing requirement for simulation studies # 1 and # 2. For the real dataset testing evaluation and simulation # 3, we adapted

the evaluation code in simulation #1.

In this Chapter, we are more focusing on part two, the implementation of target motif identification algorithm. Part one and three will be detailed in Chapter 4. At the end of this chapter. we include the performance testing results, which should give users a feeling of how fast the program runs (running space is not an issue as we discussed before, so we are not concerned with the space problem).

3.1 The Programming Development Environment

The software was developed under a UNIX operating system (Solaris) and written in C programming language by using Sun Solaris C++ (v4.2) compiler. It can also be compiled by using GNU gcc or g++ project compiler. Since it is command-line-driven program, it can be easily adapted to Microsoft window environment (WIN98 or WINNT). Borland C++ 5.0 is one of the compilers that we recommend to try. The implementation platform was Sun Workshop [1] , which is a programming environment running on Solaris. It has many wonderful features to programmers. e.g. debugging

[1]For more information, see web site below:
http://sun-www.EBay.Sun.COM:80/sundoft/Developer-products/products.html

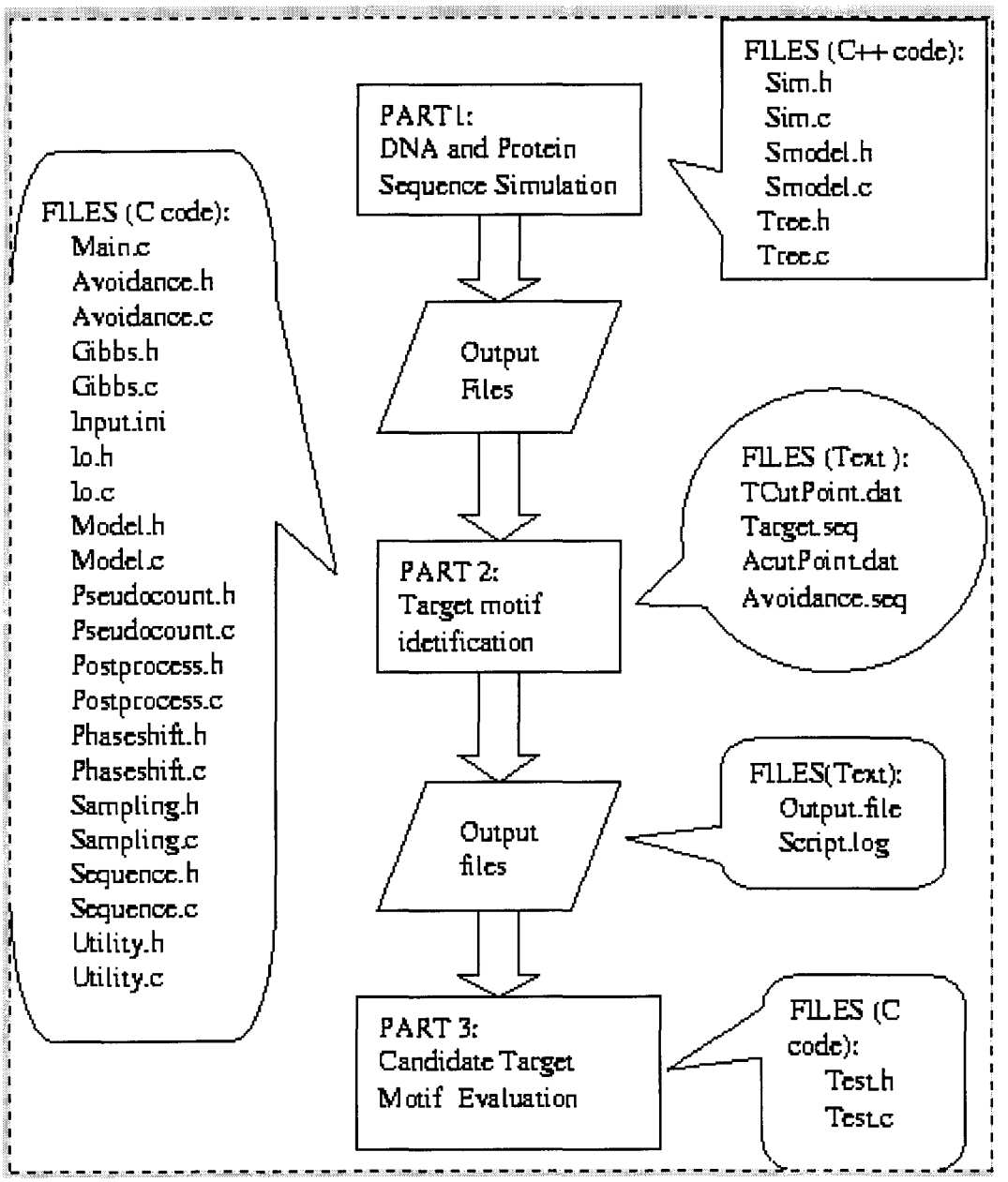

Figure 3.1: Structure of Implementation

a program, tracing memory utilization , analyzing program performance. It also supports team work.

3.2 DNA and Protein Sequence Simulator

This part (see Figure 3.1, Part 1) is implemented in C++, which contains six files, Files *sim.h* and *sim.c* manipulate the user input and output as well as how motif sequences to be embedded into individual sequences. Files *smodel.h* and *smodel.c* implement the simulation model, i.e. how the sequence is generated. Files *tree.h, tree.c* take care of phylogenical tree structure. The simulator has the ability to generate multiple sequences for both DNA and protein along a given phylogeny tree with an evolution rate on each edge. To meet our special testing purpose that we will detail in next chapter, this simulator can embed motifs into the sequences associated with each node.

3.3 Main Algorithm Implementation

Based on the logical structure shown in Figure 3.2, which is a flow-chart display of the algorithm (see Tables 2.3 through 2.10), Twenty two files (both *.h and *.c files) are created. These files can be logically divided into 9 modules, which are described in the following subsections

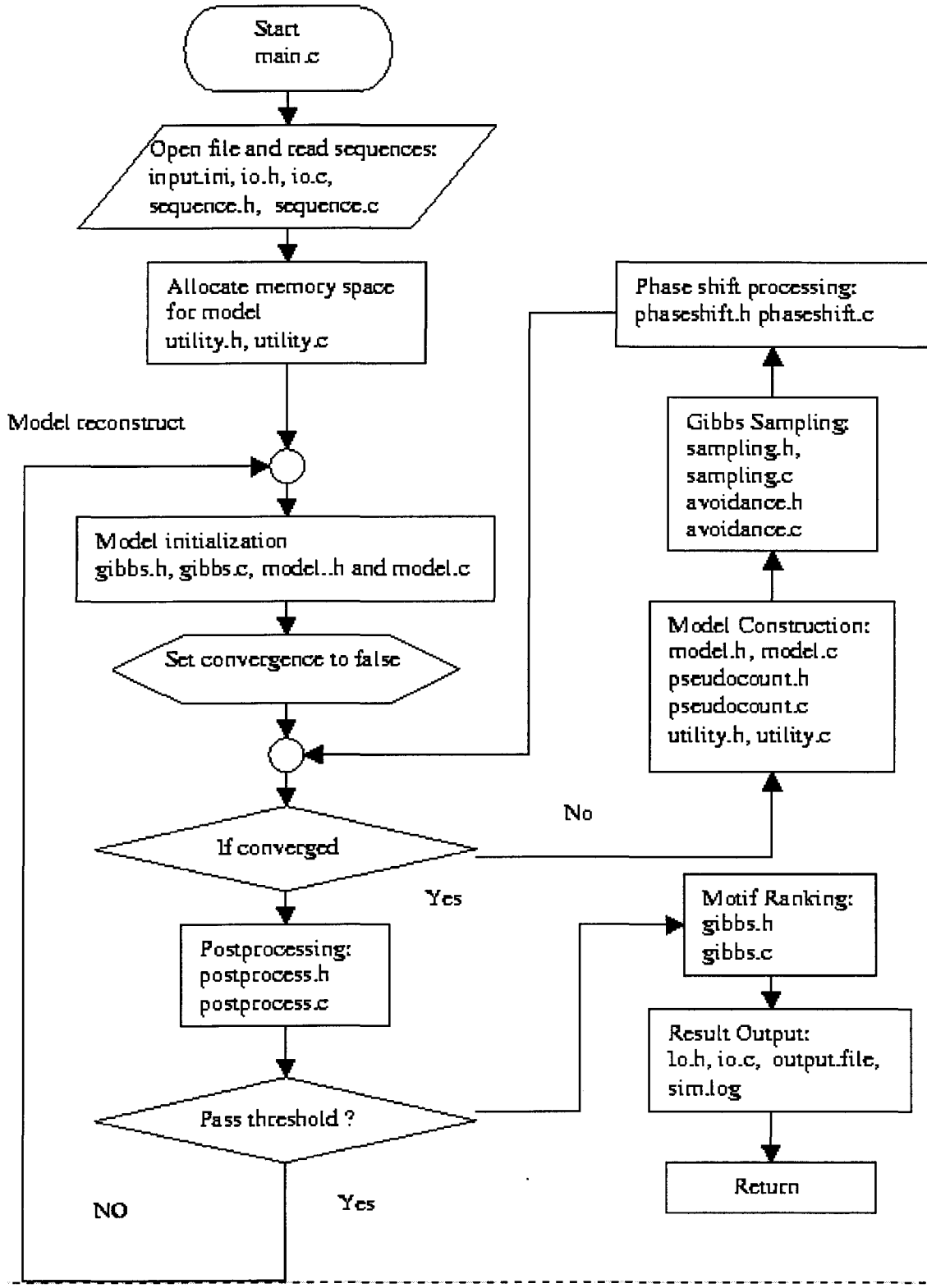

Figure 3.2: Structure of the Modified Gibbs Sampling Algorithm

3.3.1 IO Module

Parameter Input

This module includes four files: *io.h, io.c, sequence.h, sequence.c.* Files *io.h* and *io.c* take care of the user's input as well as the program's outputs. The program read in all required parameters from an input file named "input.ini". Users are allowed to specify the parameters in this file. Where appropriate, the type of the parameter and any default values are automatically used by the program if user does not specify any alternative setting.

An example of input file is given in Figure 3.3. Some notes on each of the parameters in this file are as follows:

The symbol # starts a comment line. The random seed can be any integer. If user wishes that the result be repeatable, the same seed can be used when re-running the program. The symbol tag is the switch between DNA and protein sequences, where 4 is for DNA and 20 is for protein. The expected motif length can be set between 5 to 50 bases in length as the user needs. The program may handle motif windows wider than 50, but such wide windows are not practical useful from a biologist's point of view. The separation threshold is used in the evaluation of a candidate motif against avoidance sequences, e.g. a threshold of 12 means that the best alignment of a candidate motif and any avoidance sequence substring should have fewer

```
# This is an example of input file
# seed number
76
# symbol tag DNA- 4, Protein 20
20
# Expected Motif length
20
# threshold to be considered as candidate.
12
# name of the target sequence file
target.seq
# name of the avoid sequence file
avoid.seq
# Output file name
out.file
# number of RUNS
20
# Convergence value.
100
# Symbol gap penalty
0.08
# Column Gap Penalty
1
```

Figure 3.3: An Example of The Input File for Parameter Setting.

than 12 matching bases. The higher the threshold, the lower the constraint for finding a target motif. The number of runs gives a maximum number of runs of the program that will be made: the default value is 20. The convergence parameter plays a role in terminating the program. The program will terminate if the F-value of the motif model has not increased after this number of iterations. In our experiments, it is set between 100 and 200. Symbol gap penalty is the penalty for inserting a gap when aligning a candidate motif instance to align against a motif model. It is set to 0.6 for DNA and 0.02 for protein in our experiments, but this really depends on the user's input sequences. The user should adjust this parameter accordingly to achieve optimal results. Column gap penalty is the penalty for inserting a gap into a motif model. We give 1.0 as default. Again, this is a user adjustable parameter.

Sequence Input

Files *sequence.h* and *sequence.c* handle the sequence input. FASTA format is used. Figure 3.4 shows an example of FASTA format. As it reads the sequences, the program dynamically allocates contiguous memory space according to the size of input sequences by using $calloc(size\_tnelem, size\_tsize)$ at first and then by using a series of $realloc(void * ptr, size\_tsize)$ function calls. The sequences are stored in the memory during run time (actually this

```
>Sequence Name 1: Yellow lupine
 GVLTDVQVALVKSSFEEFNANIPKNTHRFFTLVLEIAPGAKDLFSFLKGSSEVPQNNPDL
>Sequence Name 2: Vitreoscilla sp.
 MLDQQTINIIKATVPVLKEHGVTITTTFYKNLFAKHPEVRPLFDMGRQESLEQPKALAMT
```

Figure 3.4: An Example of Input Sequence Format

is the major space requirement of this algorithm).

Output Files

The output files are "output.dat" and "output.log", which contain target motif output and a script log file. An example of target motif output is shown in Figure 5.1.

3.3.2 Model Construction Module

This module includes two files, *model.h and model.c* . Three functions in these files are very important. They are *SymbolCount(), ComputeFrequence()* and *Normalization().* Function *SymbolCount()* computes a count of each symbol at each position of the motif model. Function *ComputeFrequence()* converts the number of counts of each symbol into frequency and at the same time incorporates pseudocount into it. Function *Normalization()* normalizes each symbol's frequency at each position.

For speed consideration, during model creation process, we obtain the

symbol count data by adding a count in a column from a new candidate motif instance and remove the count in a column from the motif instance of a removed sequence. We also map each symbol character in Σ into a program internal code.

3.3.3 Pseudocount Module

The motivation for making the pseudocount computation independent from the Model Module is to ease trying different ways of pseudocount calculation. In the files *pseudocount.h* and *pseudocount.c*, we include three different ways of calculating pseudocount although we only used one of them in the tests reported here.

The first one is the simple pseudocount method, which is the one we used throughout our testing study. The algorithm for this was mentioned in the previous chapter and also described in [LAB+93, pages 209 and 214]. The second method we implemented uses substitution matrix mixtures [DEKM98, pages 117 and 119]. This is not a theoretically well-founded approach, but it makes intuitive sense as a heuristic. The idea is quite natural: the pseudocount that incorporating substitution matrix term dominates if there are small numbers of training sequences and values close to the maximum likelihood estimate are obtained when the number of counts is large. However

we haven't done much testing using this method. In addition, a third pseudocount method, which is the simplest one called Laplace's rule [DEKM98, page 108], i.e. the pseudocount is constant 1, was included in our program .

Besides the ways we calculate pseudocounts, we also can scale pseudocounts by multiplying with a parameter to increase or decrease the influence of the pseudocounts to real sequence symbol counts. Generally, the degree of influence of pseudocounts to our target motif finding algorithm is not clear yet. This might be an interesting project that is worth exploring separately.

3.3.4 Avoidance Module

This module includes files *avoidance.h* and *avoidance.c.* The avoidance correlation models mentioned in Chapter 2 are constructed with order-1. The values are stored in a 2-dimensional arrays with the size of n^2, where n is the maximum number of symbols in an input sequence. Higher orders also possible, but may not be necessary since our expected motifs are usually short.

As discussed before, the product of these probabilities can get very small, and so we use logarithms. If the probability table is replaced by table of the logarithms, computing the logarithm of the probability of motif instance x is just the summation of table entries, making these simple correlation models

very fast to compute.

3.3.5 Sampling Module

This module implemented many core routines in Gibbs sampling processes. It has two files *sampling.h* and *sampling.c*. The *Cost Matrix* and *DPM Matrix* from Table 2.8 and 2.9 are implemented here. It is worth knowing that the motif model is modified if an column filled with gaps since we are introducing gaps into the model. The modifying procedure is to remove any of these columns filled with gaps during motif creation processing.

3.3.6 Other Modules

There are four other modules. They are utility module, postprocess module, phaseshift module and target motif candidate ranking module.

In utility module, files *utility.h and utility.c* contain routines to allocate and deallocate memory spaces dynamically during program running. Besides the miscellaneous routines such as mapping protein 20 characters into 7 characters based on the group similarity (see 4.5).

Postprocessing module also has two files *postprocess.h* and *postprocess.c*. Candidate target motif evaluating routines are implemented in this module. Phaseshift module has two files, *phaseshift.h* and *phaseshift.c*. The final module, target motif candidate ranking module, is to rank a list of candidate

motifs according to either F-values. It was coded in files of *gibbs.h* and *gibbs.c*.

3.4 Motif Evaluation Processing Implementation

The evaluation process, which is detailed in Chapter 4, takes the candidate target motif from our main algorithm as input to evaluate if it is the one satisfying all the conditions based on the known information from either simulated datasets or published real datasets [LAB+93, PBR89, HQH88, MVF94, OL91, NG94].

Only two files in this part: *test.h* and *test.c* see 3.1. However, based on the different evaluation processing for simulation studies and real data tests, we implemented two evaluation algorithms in these files. One is typical for simulation study #1, the other is typical for simulation study #2. For the real data testing evaluation, we adapted the simulation study #1 code.

3.5 Performance Analysis

To know how fast our program runs, we tested our program on a Sun Sparc Ultra-II workstation (300 MH CPU and 512MB RAM) on both

simulated DNA and protein datasets. Each dataset consisted of two motifs (one target) of length 20 which were embedded in N target sequences, $N = 10, 20, 30, ..., 100$, and 2 avoidance sequences of base length $L' = L - 40$, $L = 100, 200, 300, ..., 1000$ (which had post-embedding lengths L and $L - 20$, respectively). Both of the motifs are always perfectly conserved. The post-processing threshold T_p was 12, *i.e.*, 60% identity with any substring of the avoidance sequences. For each combination of N and L, five simulated datasets were constructed, the algorithm was run five times on each dataset, and the run-times of these 25 executions of the algorithm were averaged. Note that the run-times of all runs (irrespective of whether target motifs were or were not detected) were included in these averages.

The results were plotted in Figure 3.5 and Figure 3.6. Clearly, there is a linear relationship between time elapsed and the number of target sequences. The same trend for length of sequences as well. These results are consistent with the time complexity analysis in Section 2.2.7.

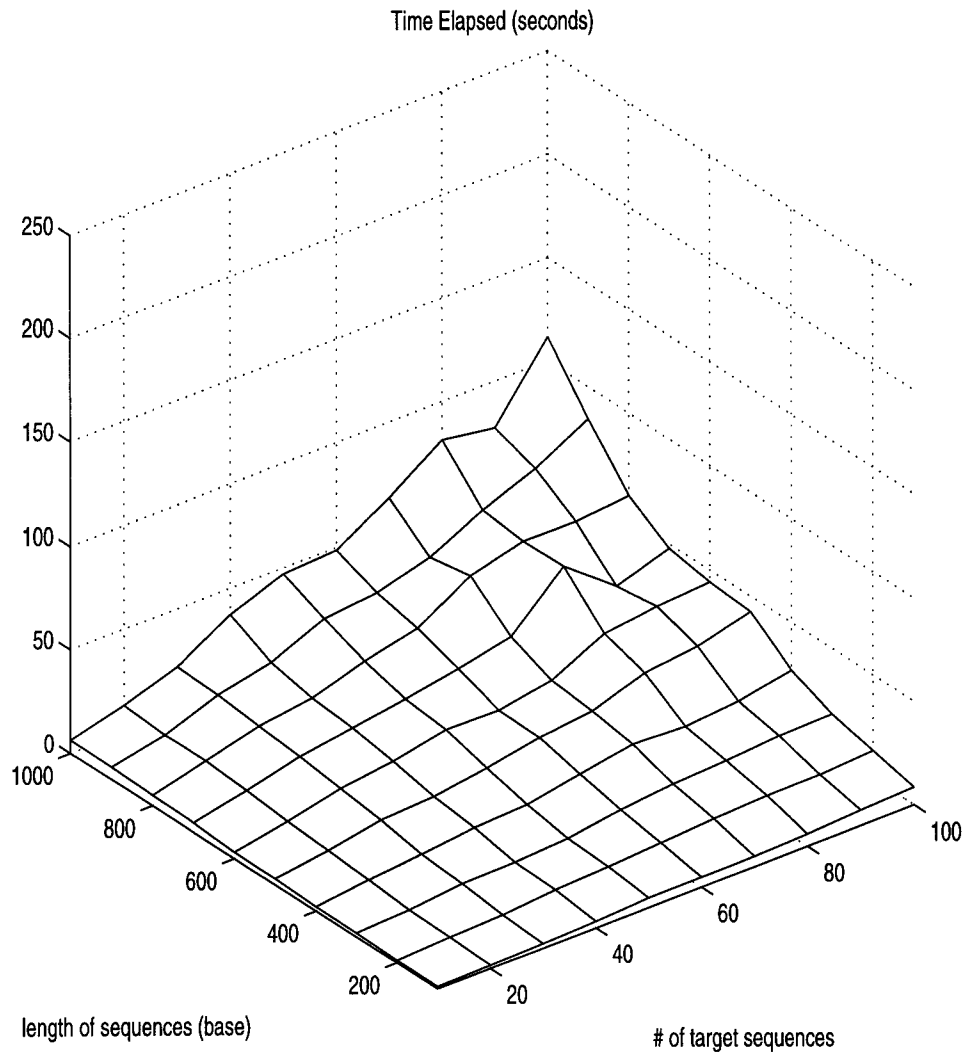

Figure 3.5: Performance on Simulated DNA Sequence Datasets

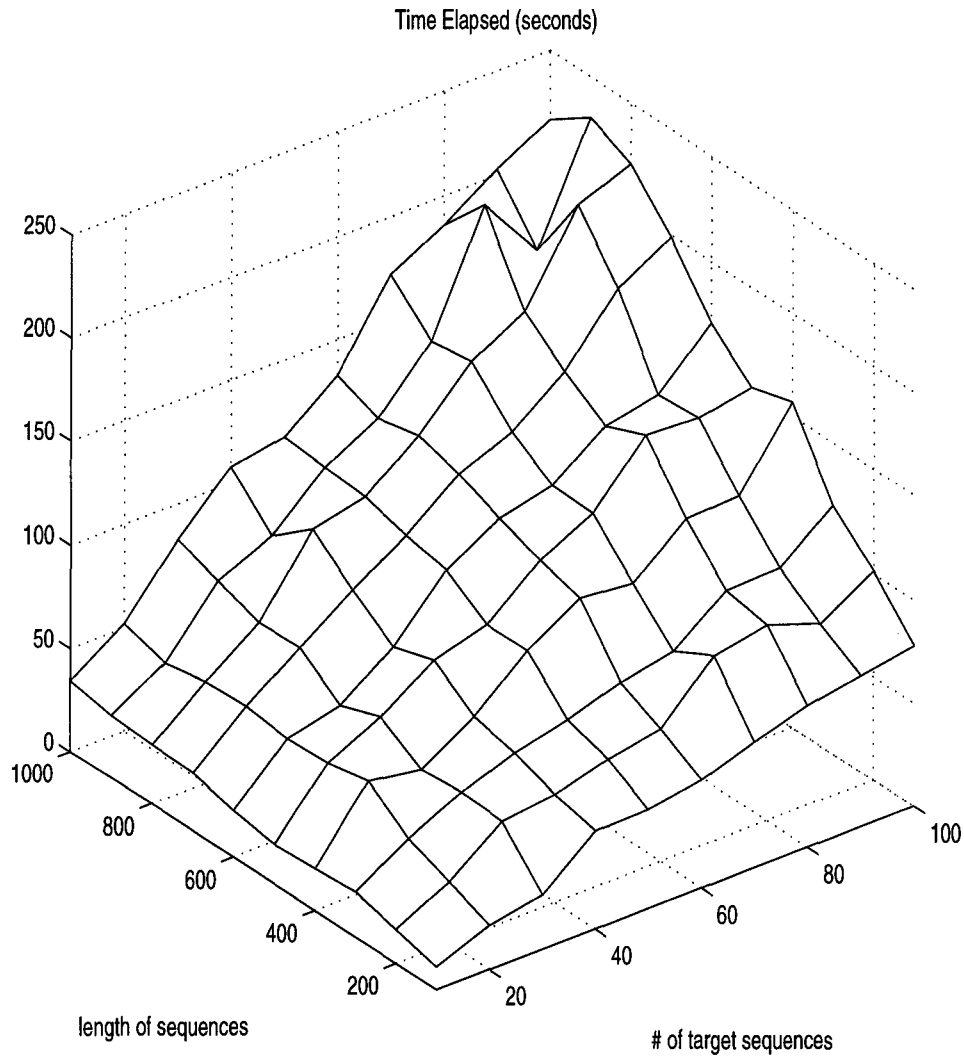

Figure 3.6: Performance on Simulated Protein Sequence Datasets

Chapter 4

Simulation Studies

Given the algorithms for solving the target motif identification problem described in the previous chapters, two questions seem particularly relevant:

1. Are methods that integrate avoidance sequence information into the motif-search process preferable to naive methods that simply iterate and post-process the results of known motif-finding algorithms?

2. If such methods are preferable, how well do they perform when confronted with datasets that contain many target motifs, *i.e.*, how many runs of the method are required to detect a significant proportion of those motifs?

To answer these two questions, we implemented both the algorithm in [RT98] and our modified version of the algorithm described in Chapter 2. The naive

approach to finding target motifs described in Table 2.6 through 2.10 was also implemented by embedding the algorithm in [RT98] in the post-processing loop described in Section 2.3. That is, first run a motif finding algorithm and then evaluate the motif to see if it satisfied some threshold value in the postprocess.

Finally, in the remainder of this Chapter, let each execution of an algorithm on a dataset be a *run* and each set of runs relative to a particular dataset be a *trial.*

In addition, by using the simulated data, we are able to answer two other important questions.

3. How well do the amino acid similarity grouping methods proposed in chapter 2 help us to detect target motifs ?

4. How well does the separation distance (see 4.5) relate to finding target motifs and how does it compare to F-values?

4.1 The Dataset Simulator

4.1.1 Principle

The studies described in this section use simulations relative to very simple simulated datasets to answer the above questions. One possible objection to

such simulations is that as the datasets are simple, the results derived from these simulations may not be relevant to real datasets. However, by virtue of being the product of the interaction of a small number of factors that are under our control, we believe that in the initial stages of an investigation of an algorithm's behavior, such results are easier to interpret and give a much more reliable and general characterization of this behavior than the traditional testing methods of applying algorithms to real data or performing simulations relative to complex datasets.

4.1.2 Method

Each dataset in the simulation consists of a set of target sequences and a set of avoidance sequences. These sequences are generated by random first as background sequences. They are completely independent of each other. This ensures us two things:

1. The background sequences diverge enough from each other, so they are unlikely have any potential target motifs

2. Each target sequence has equal weight when estimating parameters during the model creation process. This means that we don't have to worry about sequences being closely related to each other [1].

[1]The statistical model described in [LNL95, page 1157] assumes that sequences under analysis are *independent* each other. However, they found, our results also support,

• **Motif Sequences** Once we have these background sequences, the next step is to embed motifs into target sequences as well as avoidance sequences. To simulate a motif, a "seed string" is generated first and copies of that motif are created by performing a specified number of mutations at random on that motif's seed string, where this number is the product of a specified *motif dispersion rate* and the length of the motif. For example, if the requested motif length is 40 bases and evolution rate is 10%, then the mutation rate will be $40 * 10\% = 4$, i.e. among the 40 bases, there will be 4 sites to be mutated, by either substitution or insertion or deletion depending on what the user requests. and the other 36 sites will be remained unchanged.

• **Motif Mutation Rates** Throughout our simulation tests, a fixed partition of the mutation sites is used. For ungapped motif, only substitution occurs, i.e. substitution probability is 1. For gapped motif, whenever a mutation happens, a substitution has 0.60 probability, an insertion has 0.2 probability and a deletion has 0.2 probability. Note that by an appropriate choice of motif length and motif dispersion rate, one can ensure that the only motifs that can be detected in such datasets are those that are explicitly embedded in those datasets. Target motifs will only be embedded into target sequences, but avoidance motifs will be embedded into both target sequences

that this method works well even with substantial departures from this assumption. For simplicity, we are still following this assumption during simulation studies

and avoidance sequences.

4.2 Evaluating Simulated Target Motif Detection

Given datasets constructed as described above, we know the locations of all possible motifs and their degrees of conservation, and can thus not only unambiguously define what it means for an algorithm to detect a motif but more importantly determine when detection does and does not occur. Given a set of sequences S, the start and stop positions of a motif m in each sequence in S, and the start and stop positions of a candidate target motif m_c in each sequence in S, define the *average overlap of* m_c *relative to* m *in* S as the average of the overlaps of m and m_c in each sequence in S. We will say that m_c *detects* m in S if the average overlap of m_c relative to m in S exceeds a threshold T_D.

4.3 Simulation Study #1

4.3.1 Motivation

This simulation study rephrases the first question posed at the beginning of this section in terms of a potential pitfall with the naive method for target motif identification. Motif-finding algorithms are typically designed to find well-conserved motifs; hence, the motif-finding algorithms underlying the naive method should consistently prefer the best-conserved motif in the target sequences, even if that motif is a non-target motif. The question then becomes, how often does this happen and under what conditions does it start causing serious problems for the naive method (and hence validate methods like the one proposed in this thesis)?

4.3.2 Methods

Each dataset consisted of two motifs (one target) of length 20 which were embedded in 20 target sequences and 2 avoidance sequences of base length 60 (which had post-embedding lengths of 100 and 80, respectively). The non-target motif is always perfectly conserved and the target motif has a specified dispersion rate. The postprocessing threshold T_P was 12, *i.e.*, 60% identity with any substring of the avoidance sequences, and the detection threshold T_D was 10. Each trial consisted of 10 runs on a particular dataset, and the

	% Motif Dispersion	Average # Runs Target Motif Detected			
		DNA		Protein	
		Naive	Mod.	Naive	Mod.
Ungap	0	9.53	9.23	8.80	9.19
	10	9.39	9.30	8.17	8.86
	20	7.03	8.50	6.99	7.98
Gap	0	8.05	8.29	9.45	9.63
	10	6.90	5.86	7.58	8.90
	20	3.42	3.18	4.41	7.15

Table 4.1: Simulation Study #1 Results: Motif Detection.

number of runs the target motif is detected in each trial was recorded. Each trial was done on a different dataset. The naive and modified Gibbs algorithms were run for 100 trials apiece on protein and DNA sequence datasets with ungapped and gapped motifs generated relative to motif dispersion rates 0%, 10%, and 20%.

4.3.3 Results and Discussion

The results of this simulation are given in Table 4.1. Both of the algorithms perform well at low motif dispersion rates, and performance falls off as the motif dispersion rate increases (particularly in the case of gapped motifs). The modified Gibbs algorithm always performs better than the naive Gibbs algorithm in the case of ungapped motifs and gapped protein motifs; however, in the case of gapped DNA motifs, the naive Gibbs algorithm performs better

at non-zero motif dispersion rates. Two possibilities suggest themselves:

1. Despite our best efforts, some parameters have not been optimized to get the best possible performance for the modified Gibbs algorithm relative to gapped DNA motifs.

2. The modified Gibbs algorithm is sensitive to the size of the sequence-alphabet in the case of gapped motifs. This may be a product of our modifications or it may even be inherent in the original [RT98] algorithm (as that algorithm was never tested on its ability to detect known protein (let alone known DNA) motifs [RT98, Tom99]).

At this time, we cannot speak with certainty about either alternative, though the results reported in subsequent sections tend to support the latter. In any case, the results reported in this section suggest that methods that integrate avoidance-sequence information into target-motif search may be useful for detecting weakly-conserved target motifs in the presence of strongly-conserved non-target motifs.

The observant reader will have noticed that the performance of both algorithms is better for DNA than protein datasets in the case of ungapped motifs, but better for protein than DNA datasets in the case of gapped motifs. As ungapped motifs are evaluated relative to Hamming similarity and ungapped motifs are evaluated relative to edit distance in the postprocessing

loop, it seems sensible to attribute this difference in performance to the evaluation function. This suggests that the form of the evaluation function is an additional parameter that must be adjusted to optimize the performance of the modified Gibbs algorithm relative to a given dataset.

4.4 Simulation Study #2

4.4.1 Motivation

Given that algorithms like that described in this thesis which integrate avoidance-sequence information into target-motif search are preferable to naive algorithms, this simulation determines how many runs of our modified Gibbs algorithm are required to detect significant numbers of target motifs in datasets that contain multiple target motifs. This is worth knowing, as it will often be the case with real datasets that we won't know how many motifs are present and will need guidelines on how many times we must run our algorithm in order to find a significant number of these motifs.

4.4.2 Methods

Each dataset consisted of ten motifs (five target) of length 20 which were embedded in 20 target sequences and 2 avoidance sequences of base length 300 (which had post-embedding lengths of 500 and 400 respectively). Note that

this preserves the ratio of two motifs to 60 bases (one motif to 60 bases) for target (non-target) motifs that held in the first simulation study. All motifs have a common specified dispersion rate. The postprocessing threshold T_P was 12, *i.e.*, 60% identity with any substring of the avoidance sequences, and the detection threshold T_D was 10. Each trial consisted of 25 runs on a particular dataset and each trial was done on a different dataset. The modified Gibbs algorithm was run for 100 trials on protein and DNA sequence datasets with ungapped and gapped motifs generated relative to motif dispersion rates 0%, 10%, and 20%.

4.4.3 Results and Discussion

The results of this simulation are given in Table 4.2. On average, the modified Gibbs algorithm recovers more than half of the target motifs after only 10 runs and almost all of the target motifs after 25 runs under all motif dispersion rates in the case of ungapped DNA and ungapped and gapped protein motifs. The previously noted sensitivity of the algorithm when confronted with gapped DNA motifs seems to be dramatically exacerbated by having multiple motifs present. The presence of multiple motifs may also be responsible for the additional pattern present in the results that go against

	% Motif Dispersion	Average # Target Motifs Detected					
		DNA			Protein		
		# Runs			# Runs		
		5	10	25	5	10	25
Ungap	0	3.34	4.13	4.71	3.49	4.43	4.93
	10	3.19	3.94	4.56	3.38	4.32	4.90
	20	3.03	3.61	3.98	2.96	3.95	4.81
Gap	0	2.56	3.02	3.47	3.69	4.44	4.96
	10	1.48	1.95	2.61	3.28	4.23	4.81
	20	0.28	0.37	0.48	1.86	2.80	4.21

Table 4.2: Simulation Study #2 Results: Motif Detection.

the results presented in the previous section. At this time, we have no explanation for why the presence of multiple motifs should cause these patterns (however, see related results in Chapter 5).

4.5 Simulation Study #3

4.5.1 Motivation

As noted in [LAB+93], "Prior knowledge concerning amino acid relations has been used profitably in pairwise protein sequence alignment as well as in pattern construction methods" (page 213). How should such knowledge be integrated into the target motif identification process? The optimal solution would involve using such information as encoded in any one of the popular amino-acid similarity matrices, *e.g.*, PAM, Blossum. However, there is no

obvious way of integrating such matrices into the algorithm given in [RT98], and though such matrices have been integrated into the ungapped motif Gibbs Sampling algorithm given in [LAB+93] (see page 213), the authors have never explained in print how this was done.

The solution adopted here is to partition the set of 20 amino acids into m, $m < 20$, user-specified classes such that given protein sequences are mapped according to this partition into sequences over an m-symbol alphabets prior to motif search. In particular, we consider a mapping *Map7* which uses 7 symbols to encode a refined version of the commonly used non-polar / polar / charged classification of amino acids, *i.e.*, the amino acids are partitioned into the classes $\{A, I, L, M, F, W, V, Y\}$ (hydrophobic), $\{S, T, N, Q\}$ (polar but uncharged), $\{K, R, H\}$ (positively charged), $\{E, D\}$ (negatively charged), and $\{C\}$, $\{G\}$, and $\{P\}$ [Wan98]. Such mappings encode a very coarse type of amino-acid similarity, and in cases where the structure of the protein constrains how amino acids can mutate, *e.g.*, hydrophobicity must be preserved, it seems reasonable to conjecture that a mapping based on those constraints could help in detecting subtle motifs. The question addressed by the simulation in this section is, in sequences that have evolved under the constraints encoded in Map7, how well does Gibbs Sampling motif detection fare in Map7 mode as compared to unmapped mode?

	% Motif Dispersion	Average # Runs Target Motif Detected	
		Unmap	Map7
Ungap	0	8.91	8.67
	10	8.39	8.45
	20	7.53	8.08
Gap	0	9.63	8.42
	10	8.88	6.78
	20	7.36	4.83

Table 4.3: Simulation Study #3 Results. Motif detection

4.5.2 Method

The dataset simulator was reconfigured to mutate amino acids according to the following model which has parameters c and d such that $c, d \geq 0$ and $c + d = 1.0$:

$$p(x \rightarrow y) = \begin{cases} \frac{c}{(|A|-1)} & \text{if } x \text{ and } y \text{ are in the same class } A \\ \frac{d}{19-(|A|-1)} & \text{if } x \in A \text{ and } y \notin A \end{cases}$$

The balance between the values c and d expresses how predisposed substitutions are to preserve the classes in the mapping. In our simulation, we set $c = 0.75$ and $d = 0.25$, *i.e.*, amino acids are constrained to mutate largely within classes. Given this modified simulator, datasets were constructed and run against the modified Gibbs algorithm in unmapped and Map7 mode as in simulation study #1, except that each combination of mapping mode and motif dispersion rate had 50 rather than 250 associated trials.

4.5.3 Results and Discussion

The results of this simulation are given in Table 4.3. While Map7 mode does help slightly in detecting ungapped target motifs for non-zero motif dispersion rates, it performs much worse than unmapped mode for gapped target motifs. The former is consistent with [LAB+93], who found that integrating amino acid similarity information into the ungapped motif Gibbs Sampling algorithm yielded no significant improvement [LAB+93, page 213], while the latter may be another manifestation of the sensitivity noted in previous sections of the modified algorithm to small sequence alphabets when searching for gapped motifs. This is supported by the curious intermediacy of the motif detection and correlation coefficient results for Map7 mode (which effectively has a symbol alphabet of size 7) relative to those for protein and DNA datasets reported in simulation study #1 (which have symbol alphabets of size 20 and 4, respectively).

While it is disappointing that our conjecture about the utility of mapping has not been borne out by the results in this section, it is interesting to note the robustness of unmapped mode at detecting gapped motifs in the face of symbol-biased sequence datasets. This, in conjunction with the experience of [LAB+93] cited above, suggests that the modified Gibbs algorithm is (and perhaps even the original Gibbs algorithms in [LAB+93] and [RT98]

are) insensitive to prior information about the symbol-distribution in given sequence datasets. If this is so, it would make the Gibbs algorithms unique among motif-location algorithms, in that they would not depend on the availability of such prior information (in the form of nucleotide-mutation models or amino-acid similarity matrices) to perform well, and would make Gibbs algorithms ideal for finding motifs in datasets whose sequences have originated in lineages that have undergone different mutation rates. This makes further research into the effect of integrating prior symbol-distribution information into Gibbs-based motif finding algorithms very important.

4.6 Simulation Study #4

4.6.1 Motivation

Besides F-value as a target motif predicator, which has been shown to be strongly correlated with average motif overlap in simulation study #1, #2, #3. The *separation distance* (see 2.4.2), nevertheless, is particularly attractive to drug designers [Wan98].

To assess separation distance as a target motif predicator, we incorporated the calculation of this value into simulation study #1, #2, #3 and computed the correlation coefficients of the target motif overlap with this value.

4.6.2 Method

Three quantities were computed and stored for each candidate target motif created during the simulations:

1. The average overlap;

2. The F-value;

3. The separation distance

The correlation of each pair of these three quantities was assessed using Pearson correlation coefficients [MMH90, Chapter 6], whose values range from -1 (strong negative correlation) to 0 (no correlation) to 1 (strong positive correlation). If the relationship between the variables is linear, high ($> .5$) coefficient values for pairs of quantities suggest that the values of each of the quantities involved is a good predictor of value of the other quantity. The sample size is 5000 in simulation study # 1 and 2500 in simulation study # 2 and 5000 in simulation #3.

4.6.3 Result and Discussion

The correlation coefficients are presented in parts (a), (b) and (c) of Table 4.4, that covers various situations, which includes DNA and protein, gapped and ungapped, in unmapped and Map7 mode. The results show that F-value

is almost always more strongly correlated with (and is hence always a better predictor of) average overlap than separation distance. Though the strength of these correlations always decreases with increasing motif dispersion rate, the decrease is particularly dramatic in the case of DNA motifs (especially gapped DNA motifs). This may be another manifestation of the apparent sensitivity noted above of the modified Gibbs algorithm to the size of the sequence alphabet. In any case, the strong correlation of F-value with average overlap suggests that a reasonable strategy for detecting target motifs in sequences would be to sort candidate target motifs by F-value and then to determine if any of the highest-scoring candidate target motifs occur in the same regions of the given sequences (the repetition being necessary by virtue of imperfect correlation). This is essentially the strategy derived empirically in and used throughout [LAB+93] to detect motifs (see caption of [LAB+93, Figure 3]). The separation distance can be used as a reference parameter when the target motifs are DNA strings with relative high dispersion rates.

	% Motif Dispersion	Correlation with Average Overlap			
		DNA		Protein	
		F	S	F	S
Ungap	0	0.9701	0.7967	0.9805	0.7231
	10	0.9499	0.8023	0.9701	0.7117
	20	0.4914	0.5191	0.9550	0.7396
Gap	0	0.9778	0.9245	0.9812	0.8678
	10	0.7323	0.7446	0.9229	0.8125
	20	0.4514	0.2898	0.8359	0.7401

(a)

	% Motif Dispersion	Correlation with Average Overlap			
		DNA		Protein	
		F	S	F	S
Ungap	0	0.9821	0.9412	0.9868	0.9688
	10	0.9152	0.9378	0.9748	0.9217
	20	0.9445	0.9501	0.9498	0.8265
Gap	0	0.9937	0.9694	0.9900	0.9478
	10	0.6403	0.7348	0.8849	0.7989
	20	0.3916	0.4798	0.8030	0.6265

(b)

	% Motif Dispersion	Correlation with Average Overlap			
		Unmap		Map7	
		F	S	F	S
Ungap	0	0.9721	0.6638	0.7973	0.7608
	10	0.9698	0.6550	0.7857	0.7689
	20	0.9572	0.7064	0.7245	0.7712
Gap	0	0.9812	0.8678	0.8113	0.8814
	10	0.9238	0.8319	0.6054	0.7580
	20	0.8321	0.7195	0.3996	0.3801

(c)

Table 4.4: Correlation Coefficients Results: a) in Simulation Study #1. b) in Simulation Study #2. c) Simulation Study #3. F – represent F-values and S – represents the separation distances

Chapter 5

Experiments on Real Data Sets

5.1 Motivation

The previous results in this thesis suggest that our algorithm is good at identifying target motifs in simulated datasets under a variety of conditions. However, the simulator used to create the examined datasets is admittedly simplistic and the produced sequences and motifs probably do not exhibit crucial characteristics of real sequences and motifs. On the other hand, the real datasets we used are much more complex. The experiments in this section were designed to address a relevant question:

How well does our modified Gibbs algorithm perform on real datasets?

5.2 Method

5.2.1 Real Dataset Archives

Real data sets were selected based on some proteins sequences, which contain well-supported conserved regions (motifs). After a thorough literature survey, five sets were chosen. These are cytosine methyltransferases[PBR89], protein kineases [HQH88, MVF94], lipocalins[LAB+93], cyclins [OL91] and acetyltransferase [NG94]. We retrieved these sequences from NCBI database.

5.2.2 Target Motif Dataset Creation

Following method is used to construct target and avoidance sequences for our testing purpose. We remove one or more protein sequences from a set of retrieved sequences to form an avoidance sequence set. The remaining sequences are the target sequence set. Choose a well-known motif in the target sequence set as a target motif and randomly permute the amino acids of the corresponding segments in all avoidance sequences; this effectively "erases" the occurrences of the target motif in the avoidance sequences. To evaluate our program against real datasets, we adapt the evaluation code in simulation study #1 here, performed 50 trials and each trial consists of 10 runs both in unmapped and Map7 modes.

5.2.3 Parameter Setting

It worth of knowing that some parameters used were consistant on all the real dataset tests. They are: convergence value 100, single gap penalty 0.02 and column gap penalty 1.0 in unmapped mode; convergence value 100, single gap penalty 0.4 and column gap penalty 1.0 in Map7 mode. The pseudocount setting is the same for both unmapped and Map7 modes. The other parameters varied from datasets.

5.3 Test Results

5.3.1 The Cytosine Methyltransferase Experiment

As a starting point for our analysis, we chose a dataset with relatively more conserved regions (i.e. motifs) and window sizes close to the ones in our simulation study (i.e. 20 amino acids).

DNA methyltransferases (MTases) recognize specific nucleic acid sequence patterns in their targets and transfer methyl groups from the donor *S-adenosylmethionine (SAM)* to adenine or cytosine residues [PBR89]. They contain 5 highly conserved regions of which we considered 4. The motif window size varies from 20 to 24 amino acids. This dataset contains 4 target proteins and 1 avoidance protein with 398 amino acids in length on average.

Sequence Codename	Sequence Length	Motifs			
		I	IV	VI	VIII
EcoRII	477	98 – 117	177 – 200	226 – 245	274 – 293
HhaI	327	14 – 33	72 – 95	112 – 131	152 – 171
MspI	348	37 – 56	95 – 118	136 – 155	176 – 195
Phi3T	443	6 – 25	69 – 92	142 – 161	182 – 201
BspRI (*)	396	32 – 51	119 – 142	158 – 177	198 – 217
Motif Length		20	24	20	20

Table 5.1: The Cytosine Methyltransferase Motif Data Set [PBPR89, Figures 2 and 3]. The sequences BspRI, HhaI, MspI, and EcoRII are components of Type II bacterial restriction modification systems, and sequence Phi3T occurs in *Bacillus* phages. The starred (*) sequence was selected as the avoidance sequence when this dataset was transformed into target motif datasets. See main text for an explanation of terms.

Table 5.1 shows the start and end positions in each sequence as well as motif length.

We have run our algorithm against all the target motifs in both unmapped mode and Map7 mode. The results are given in Table 5.2, which are very encouraging. We successfully detected 3 out of 4 target motifs in unmapped mode with relatively high detection rates. For instance, in the case of Motif IV, which is one of the homologue with 10 residues matching inside the target motif. We were able to find the target motif most of the 10 runs on average. The detection rates (Average # runs motif detected) are 8.26 and 8.50 by using unmapped and Map7 mode (see Table 5.2), respectively. The Map7 mode gives a slightly better performance on this motifs. Besides, motif VI and VIII are detected with detection rates at 3.78 and 5.26, respectively, in

Motif Code	Average # Runs Motif Detected	
	Unmap	Map7
I	0.42	0.00
IV	8.26	8.50
VI	3.78	1.30
VIII	5.26	0.16

Table 5.2: Cytosine Methyltransferase Motif Detection Results, Average # Runs Motif Detected generated from 50 trials. Each trials contains 10 runs. For example, if a motif is detected, then we counts 1. If we detect 8 times in 10 runs, the # Runs Motif Detected is 8.00. The values given in this table are the averages among 50 trials.

unmapped mode. The example of target motif identified is presented below in Figure 5.1, which is motif VIII in unmapped mode.

However, we also notice that for motif I and most of the tests in Map7 mode, it shows low detection rates or even failed to find one (e.g. motif I in Map7 mode). This pattern occurred in all the datasets in this chapter. To interprete this performance drop off on some target motifs, especially in Map7 mode, we conducted further experiments and will discuss these results in Chapter 6.

5.3.2 The Protein Kinase Experiment

Protein kinases are enzymes that transfer a phosphate group from a phosphate donor onto an acceptor amino acid in a substrate protein [HT91].

```
Expected Motif Length 20;  Converging Value = 100

start pos        motif                          seq name

    275         IDGKHFL-PQHRERIVLVGF            EcoRII
    152         LNALDYGIPQKRERIYMICF            Hhai
    176         LDASHFGIPQKRKRFYLVAF            MSPI
    182         LNSKFFNVPQNRERVYIIGI            PHI3T
                        ** * *
F = 19.000967    motif    length = 20
The consensus sequence is: DAKHFGIPQKRERIYLIGF

# of iteration:  291
Time taken on Average: 54.96 seconds
```

Figure 5.1: Cytosine Methyltransferase Target Motif-VIII Alignment

They comprise of catalytic domain which ranges from 250 to 300 amino acids residues. Catalytic domain amino acid residues are conserved throughout the entire protein kinase family. To be representative, the sequences selected have a broad phylogenetic distribution and a multiple alignment of these sequences has been reported in [MVF94]. Table 5.3 details the target motifs.

Compared to cyclin methyltransferases, the kinase data set is less degree of conservation and has shorter size of motif length. We select 10 proteins as target sequences and 2 as avoidance sequences. The testing results are shown in Table 5.4. We successfully detected motif IV in unmapped mode

with detection rate at 8.06. And we detected motif VI and VII with lower rates at 1.20 and 3.70 in unmapped mode also. But we failed in Map7 mode except for motif IV with very low detection rate 1.10. Figure 5.2 is an example of our program's output on Motif IV.

```
Expected Motif Length 20;        Converging Value = 100

start pos       motif                          seq name

   113          FEYLHSLDLIYRDLKPENLL           CAPK
   118          IAYCHSHRILHRDLKPQNLL           CD28
   131          LLFLHSQSIVHLDLKPANIL           CMOS
   109          MAYVERMNYVHRDLRAANIL           CSRC
   115          MNYLEDRRLVHRDLAARNVL           EGFR
   104          VDYIHRQGIIHRDIKTENIF           HSVK
   111          ILFMHKMRVLHLDLKPENIL           MLCK
   183          MDFLASKNCVHRDLAARNVL           DGM
   110          VRYLHALGITHRDLKPENLL           SKH
   109          MDYLHAKNIIHRDMKSNNIF           RAF1
                *    *
F = 21.579072   motif length = 20
The consensus sequence is: DGMDYLHSKNIVHRDLKPEN

# of iteration:  168
Time taken on Average: 41.13 seconds
```

Figure 5.2: Protein Kinase Target Motif-IV Alignment

Sequence Codename	Sequence Length	Motifs			
		I	IV	VI	VII
CAPK	255	8 – 17	113 – 132	160 – 169	178 – 187
CD28	293	8 – 17	118 – 137	168 – 177	187 – 196
CMOS	286	8 – 17	131 – 150	184 – 193	202 – 211
CSRC	257	8 – 17	109 – 128	160 – 169	178 – 187
EGFR	264	8 – 17	115 – 134	167 – 176	185 – 194
HSVK	263	8 – 17	104 – 123	156 – 165	174 – 183
MLCK	261	8 – 17	111 – 130	162 – 171	180 – 189
PDGM	340	8 – 17	183 – 202	235 – 244	253 – 262
PSKH	263	8 – 17	110 – 129	164 – 173	182 – 191
RAF1	261	8 – 17	109 – 128	161 – 170	182 – 191
VFES (*)	271	23 – 32	127 – 146	179 – 188	197 – 206
WEE1 (*)	280	8 – 17	117 – 136	165 – 174	183 – 192
Motif Length		10	20	10	10

Table 5.3: Characterization of the Protein Kinase Motif Data Set ([MVF94, Figure 2]; see also [HQH88, Figure 1]). The sequences are bovine cardiac muscle (CAPK), *Saccharomyces cerevesiae* CD28 (CD28), human oncogenic protein CMOS (CMOS), chicken oncogenic protein CSRC (CSRC), human EGF receptor (EGFR), herpes-simplex virus kinase (HSVK), rat skeletal muscle (MLCK), mouse PDGF receptor(PDGM), Hela cell (PSKH), human oncogenic protein RAF1 (RAF1), feline sarcoma virus oncogenic protein (VFES), and *Schizosaccharmyces pombe* WEE1 (WEE1). The starred (*) sequences were selected as the avoidance sequences when this dataset was transformed into target motif datasets. See main text for an explanation of terms.
Note that the sequences in this dataset are a eukaryotic subset of the sequences given in [HQH88]. The motif-codes in this table are from [MVF94]; the correspondence with the motif-codes in [HQH88] is I $\rightarrow$ I, IV $\rightarrow$ VI, VI $\rightarrow$ VIII, and VII $\rightarrow$ IX.

Motif Code	Average # Runs Motif Detected	
	Unmap	Map7
I	0.16	0.00
IV	8.06	1.10
VI	1.20	0.00
VII	3.70	0.00

Table 5.4: Protein Kinase Motif Detection Results

5.3.3 The Lipocalin Experiment

The lipocalin protein sequence dataset was used in [LAB+93] as one of the most difficult test cases since it contains two weak sequence motifs, centered on the generally conserved residues -Gly-X-Trp- and -Thr-Asp-. These two motifs are recognized from structural comparisons [CNJ90, LS87]. This dataset contains 4 target and one avoidance proteins. The sequences are 184 residues on average.

After running our algorithm against this dataset, we summarized the results in Table 5.5. Our algorithm perform well in unmapped mode but not in Map7 mode. An example of the target motif A generated by our algorithm is shown in Figure 5.3, which is exactly the same pattern as shown in [LAB+93].

Sequence Codename	Sequence Length	Motifs	
		A	B
ICYA_MANSE	189	17 - 32	104 - 119
LACB_BOVIN	178	25 - 40	109 - 124
BBP_PIEBR	189	31 - 46	115 - 130
RETB_BOVIN	183	14 - 29	105 - 120
MUP2_MOUSE (*)	180	27 - 42	109 - 124
Motif Length		16	16

Table 5.5: Characterization of the Lipocalin Motif Data Set [LAB+93, Figure 4]. The sequences and their associated SwissProt database codes are *Manduca sexta* insecticyanin (ICYA_MANSE), bovine β-lactoglobulin (LAVB_BOVIN), *Pieris brassicae* bilin-binding protein (BBP_PIEBR), bovine plasma retinol-binding protein (RETB_BOVIN), and mouse major urinary protein 2 (MUP2_MOUSE). The starred (*) sequence was selected as the avoidance sequence when this dataset was transformed into target motif datasets. See main text for an explanation of terms.

Motif Code	Average # Runs Motif Detected	
	Unmap	Map7
A	5.54	0.00
B	2.82	0.08

Table 5.6: Lipocalin Motif Detection Results.

```
Expected Motif Length 18;        Converging Value = 200

start pos       motif                              seq name

   17           FDLSAFAGAWHEIAKLP                  ICYA_MANSE
   25           LDIQKVAGTWYSLAMAA                  LACB_BOVIN
   31           FDWSNYHGKWWEVAKYP                  BBP_PIEBR
   14           FDKARFAGTWYAMAKKD                  RETB_BOVIN
                 *     * *   *
F = 16.323679   motif length = 18
The consensus sequence is: FDISAFAGTWYEIAKAP

# of iteration:  794
Time taken on Average: 82.9 seconds
```

Figure 5.3: Lipocalin Protein Target Motif A Alignment

5.3.4 The Cyclin Experiment

Cyclins are universal cell cycle regulators. Cycline proteins share strong sequence similarities. This dataset contains 198 residues on average which includes four homologue. By using this protein set, we are able to see how our program performs on motifs with longer window sizes (24 to 30 amino acids).

The test dataset contains 7 target and 3 avoidance proteins. 4 motifs has been reported in these sequences with various degree of conservation [OL91] (see Table 5.7).

The running results are summarized in Table 5.8. That tells us that our

Sequence Codename	Sequence Length	Motifs			
		I	II	III	IV
CLAM_B	189	1 - 35	44 - 59	75 - 100	137 - 157
HUMAN_B1	190	1 - 35	44 - 59	75 - 100	137 - 157
DROS_A	192	1 - 35	44 - 59	75 - 100	138 - 158
Soy	191	1 - 35	44 - 59	75 - 100	137 - 157
M-CYL2	197	1 - 35	44 - 59	75 - 100	141 - 161
H-CYC_E	207	1 - 35	45 - 60	76 - 101	148 - 168
pucl	196	1 - 35	44 - 59	77 - 102	140 - 160
CLN1 (*)	219	1 - 35	44 - 59	100 - 125	163 - 183
STAR_B (*)	191	1 - 35	44 - 59	75 - 100	137 - 157
Carrot (*)	196	1 - 35	44 - 59	75 - 100	141 - 161
Motif Length		35	16	26	21

Table 5.7: Characterization of the Cyclin Core Region Motif Data Set [OL91, Figure 2]. The sequences are clam cyclin B (CLAM_B), human cyclin B (HUMAN_B1), *Drosophila* cyclin A (DROS_A), soy cyclin A (Soy), mouse CYL2 (M-CYCL2), human cyclin E (H-CYC_E), *Saccharmyces cerevesiae* Cln1 (CLN1), *Schizosaccharomyces pombe* pucl (pucl), starfish cyclin B (STAR_B), and carrot cyclin B (Carrot). The starred (*) sequences were selected as the avoidance sequences when this dataset was transformed into target motif datasets. See main text for an explanation of terms.

algorithm works very well for longer target motifs. Especially on motifs I and III, the detection rates are 8.96 and 8.0 in unmapped mode, respectively. We also notice that the program performs well on motif I and II in Map7 mode, which implies that it prefer longer motifs. Figure 5.4 shows an example of the alignment on motif III.

Very strange, zero detection rate on motif IV in both unmapped and Map7 mode. In Chapter 6, we will show some of the characteristics of this motif and give our explanation.

Motif Code	Average # Runs Motif Detected	
	Unmap	Map7
I	8.96	6.70
II	6.44	5.14
III	8.00	0.86
IV	0.00	0.00

Table 5.8: Cyclin Motif Detection Results.

5.3.5 The Acetyltransferase Experiment

Acetyltransferases is another subtle motif dataset. It has been reported in [NG94]. This is a diverse set of enzymes including S- and N-acyltransferases. Closely related sequences were removed such that no two sequences had a PAM120 similarity score above 100 [NG94];

```
Expected Motif Length 26;        Converging Value = 100

start pos    motif                          seq name

 75      AYTKKEILEMEQHILKKLNFSFGRPL      Atlantic surf clam
 75      TYTKHQIRQMEMKILRALNFGLGRPL      cyclin B1 - human
 75      SYTKAQVLRMEQVILKILSFDLCTPT      Drosophila cyclin A
 75      AYTHEQILAMEKTILNKLEWTLTVPT      Soy mitotic cyclin
 75      SVKPQELLEWELVVLGKLKWNLAAVT      M-CYL2 cyclin 2
 76      ACSGDEILTMELMIMKALKWRLSPLT      cyclin E - human
 77      IYAEDLFIRMERHILDTLDWDISIPT      CG1P_SCHPO CYCLIN
      *       *
F = 24.054798   motif length = 26
The consensus sequence is: NAYTKDEILEMELHILKKLKWDLGRP

# of iteration:  290
Time taken on Average: 83.83 seconds
```

Figure 5.4: Cyclin Target Motif-III Alignment

Sequence Codename	Sequence Length	Motifs	
		A	B
RIMI_ECOLI	148	66 – 88	107 – 124
STA_STRLA	189	107 – 129	148 – 165
ATDA_HUMAN	171	91 – 113	132 – 149
iTTR_PSESY	177	93 – 115	133 – 150
PUAC_STRLP	199	125 – 147	163 – 180
IAAT_AZOBR	153	77 – 99	118 – 135
STA_ECOLI (*)	174	86 – 108	127 – 144
AAC1_PSEAE (*)	177	108 – 130	149 – 166
AAC6_KLEPN (*)	201	116 – 138	158 – 175
Motif Length		23	18

Table 5.9: Characterization of the Acetyltransferase Motif Data Set [NG94, Figure 3]. The sequences are denoted by their associated SwissProt database codes. The starred (*) sequences were selected as the avoidance sequences when this dataset was transformed into target motif datasets. See main text for an explanation of terms.

This dataset consists of 6 target proteins and 3 avoidance proteins with target sequence length at 172 residues on average and two motifs to be targeted. Table 5.9 shows more details.

Running our algorithm against this dataset, results are shown in Table 5.10. It is interesting that we failed to detect both motif A and B in unmapped mode. By using the Map7 approach, we are able to detect the target motif A with detection rate of 6.32 on average in 50 trials. On the same data, we had a much lower detection rate (0.08) using the unmapped mode. This implies that amino acid similarity grouping approach, i.e. Map7 in this case, does help in some case to detect target motifs. An example of the detected target motif A using Map7 mode is presented in Figure 5.5. The sequences are mapped from 20 characters into 7 characters as we have seen.

Dataset	Motif Code	Average # Runs Motif Detected	
		Unmap	Map7
Acetyltransferase	A	0.08	6.32
	B	0.00	0.02

Table 5.10: Acetyltransferase Motif Detection Results.

```
Expected Motif Length 18;        Converging Value = 200

start pos       motif                          seq name

   66           ASAAAEPEASKSGAGKAA             E. COLI
   107          EEAEAAPGKKGKGAGKAA             STA_STRLA
   91           EEAAAASEAKGAGAGSEA             ATDA_HUMAN
   93           SKAAAAPSAKGKGAGKSA             TTR_PSESY
   125          ASAGASPEKSGKGAGSAA             PUAC_STRLP
   77           KKAAASPSAKGGSAGKKA             IAAT_AZOBR
                  * *         **  *
F = 8.903478    motif length = 18
The consensus sequence is: ASAAAAPEAKGKGAGKAA

# of iteration:  349
Time taken on Average: 6.28 seconds
```

Figure 5.5: Acetyltransferase Target Motif A Alignment

5.4 Summary

Our algorithm can detect motifs in many real proteins with different degree of conservation, different motif lengthes and different numbers of input target sequences. unmapped mode shows better performance than Map7 mode most of the time. However, Map7 mode does function better on some of the real datasets. For example, of Acetyltransferase Target motif A shows high detection rate in Map7 mode but very low rate in unmapped mode Table 5.10.

Like many multiple sequence alignment algorithms, our algorithm is sensitive to its parameters. The value of these parameters varied in different

datasets. The value of the single gap penalty, threshold and motif window size parameters as well as pseudocount setting seem to make a big difference in performance. For this reason among others, one of the reasons, our algorithm was not successful on all the target motif tests as these tests used consistent parameter settings. Systematically exploring the effects of these parameters on our algorithm performance would be very useful and a good project for the future. Questions raised by our experiences includes:

1. Does the alphabet size is really affect on the algorithm performance ?

2. To what degree of pseudocounts influence motif detection relative to the size of alphabet and input sequence size ? For instance, should we increase the influence of pseudocounts when the input sequence size is small ?

3. What are the optimal parameter values for the single gap penalty and the column gap penalty on different size of alphabet sequences that are relative to different sequence types and mapping modes?

With respect to the last question, Gusfield in [Gus97, page 312-321] described parametric sequence alignment, which may be adapted to this study.

Chapter 6

Discussion

As we have seen in Chapter 4 and 5, the simulation tests prove that our program is able to quickly detect various kinds of target motifs. However, on real datasets, in many cases, it still has difficulty of identifying target motifs in both unmapped and Map7 mode by even trying different parameters. What other factors could cause this discrepancy of the performance of our algorithm on simulated and real datasets? In order to interpret this and find out the main reason causing the detection failure, as well as to provide some clues for future improvement to our algorithm, we analyzed the real datasets based on four types of characteristics.

6.1 Characteristics

Each motif was characterized by the following four characteristic quantities:

1. Motif length.

2. *Motif Distinctness*: The degree of the distinctness of that motif from the background portions of its sequences. It was measured using Kullback-Leibler distance. Given an ungapped motif of length W in a set of N sequences over an alphabet Σ, let Q be the frequency of occurrence of the symbols in Σ in the instances of that motif in the sequences, Q_i, $1 \leq i \leq W$, be the frequency of occurrence of the symbols in Σ at position i in the instances of that motif in the sequences, and P be the "background" frequencies of occurrence, *i.e.*, the frequency of occurrence of the symbols in Σ in all non-motif portions of the sequences. Two distances were computed: $H(Q||P)$, which is a measure of the conservation of the overall symbol-distribution in the motif relative to the background (Ovr), and $(\sum_{i=1}^{W} H(Q_i||P))/W$, which is a measure of the average conservation of the symbol-distribution for each position of the motif relative to the background (Avg). Each of these quantities was in turn computed relative to the original protein sequences and the sequences as transformed under the $Map7$ mapping described simulation experiment 3 in Chapter 4.

3. *Degree of Target-Sequence Interference*: Unlike the simulated datasets examined in previous sections, in which the occurrences of motifs in

sequences were unique by virtue of the manner in which these datasets were constructed, real datasets may have multiple copies of a particular motif in each sequence, each with a differing degree of conservation. Call such partially-conserved copies of a motif *ghosts*. Given that our criterion of motif-detection measures overlap of a candidate target-motif with only one of these copies and the stochastic nature of our algorithm makes it possible that our algorithm may accidentally lock onto one of the ghosts (especially if the ghost is as conserved as our known copy), the detection criterion can be misled to reject what is otherwise the correct target motif. To this end, we computed two measures of the similarity of ghosts in the avoidance sequences to known motifs,

(a) Average minimum difference between consensus string / known motif and consensus string / ghost (CAMD(m)):

$$\text{CAMD}(m) = (\sum_{i=1}^{N} d(c, m_i) - \max_{s \in s(T_i), s \neq m_i} d(c, s))/N$$

(b) Average minimum difference between known motif and ghost (GAMD(m)):

$$\text{GAMD}(m) = (\sum_{i=1}^{N} W - \max_{s \in s(T_i), s \neq m_i} d(m_i, s))/N$$

where m is the motif, c is the consensus string associated with m, m_i is the occurrence of m in the ith target sequence, $d(x, y)$ is the Hamming similarity of strings x and y, W is the length of m, and $s(T_i)$ is the set of all substrings of length W in the ith target sequence. These quantities were normalized and expressed as a percentage of the length of their associated motif. Each of these quantities was in turn computed relative to the original protein sequences and the sequences as transformed under the Map7 mapping described in Section 4.5.

4. *Degree of Avoidance-Sequence Interference*: The ghosts alluded to above may also exist in avoidance sequences. In this case, even though the known copy of the motif has been erased, a ghost that is sufficiently conserved might cause the post-processing loop to reject what is otherwise the correct target motif. To this end, we computed two measures of the similarity of ghosts in the avoidance sequences to known motifs,

$$\mathrm{CMD}(m) = (\sum_{i=1}^{N} d(c, m_i) - \max_{s \in s(A_i)} d(c, s))/N$$

where $s(A_i)$ is the set of all substrings of length W in the ith target sequence. This quantity was normalized and expressed as a percentage of the length of its associated motif. This quantity was in turn computed relative to the original protein sequences and the sequences as transformed under the Map7 mapping described in simulation experiment 3

Motif Characteristic		Motifs			
		I	IV	VI	VIII
Motif Length		20	24	20	20
Relative Entropy	Ovr/Unmap	0.6483	0.3338	0.3474	0.2405
	Ovr/Map7	0.4105	0.1638	0.0461	0.0684
	Avg/Unmap	3.0129	3.1948	2.9953	3.0322
	Avg/Map7	1.8117	1.9377	1.8375	1.6545
Target Ghosts	CAMD/Unmap	33.75	42.71	40.00	40.00
	CAMD/Map7	25.00	30.21	35.00	31.25
	GAMD/Unmap	73.75	78.12	76.25	75.00
	GAMD/Map7	76.25	79.17	75.00	72.50
Avoidance Ghosts	CMD/Unmap	25.00	29.17	25.00	25.00
	CMD/Map7	55.00	50.00	55.00	55.00

Table 6.11: Characterization of the Cytosine Methyltransferase Motif Data in Chapter 4.

The values of these quantities mentioned above for each motif considered here are given in Tables 6.15 through 6.11. These quantities were subsequently correlated with the detect ability of each motif using the Pearson correlation coefficient described in Section 4.6.2.

6.2 Correlation Anaylysis Between Characteristic Quantities

Consider first the correlation coefficients given in Table 6.16. Our initial experience with the motifs in the acetyltransferase and lipocalin datasets

Motif Characteristic		Motifs			
		I	IV	VI	VII
Motif Length		10	20	10	10
Relative Entropy	Ovr/Unmap	0.7173	0.2719	0.4417	0.5854
	Ovr/Map7	0.4378	0.0636	0.1847	0.1966
	Avg/Unmap	2.6640	2.8200	2.7382	2.8062
	Avg/Map7	1.6407	1.4483	1.5438	1.4573
Target Ghosts	CAMD/Unmap	30.00	36.50	28.00	26.00
	CAMD/Map7	12.00	23.50	4.00	9.00
	GAMD/Unmap	66.00	77.00	65.00	66.00
	GAMD/Map7	66.00	75.50	65.00	68.00
Avoidance Ghosts	CMD/Unmap	40.00	25.00	40.00	30.00
	CMD/Map7	50.00	55.00	50.00	60.00

Table 6.12: Characterization of the Protein Kinase Motif Data Set

Motif Characteristic		Motifs	
		A	B
Motif Length		16	16
Relative Entropy	Ovr/Unmap	0.4142	0.3932
	Ovr/Map7	0.1291	0.1216
	Avg/Unmap	3.0217	2.9573
	Avg/Map7	1.3082	1.4161
Target Ghosts	CAMD/Unmap	39.06	37.50
	CAMD/Map7	21.88	20.31
	GAMD/Unmap	75.00	75.00
	GAMD/Map7	71.88	71.88
Avoidance Ghosts	CMD/Unmap	18.75	18.75
	CMD/Map7	56.25	50.00

Table 6.13: Characterization of the Lipocalin Motif Data Set

Motif Characteristic		Motifs			
		I	II	III	IV
Motif Length		35	16	26	21
Relative Entropy	Ovr/Unmap	0.1784	0.3917	0.0647	0.2809
	Ovr/Map7	0.0742	0.1782	0.0157	0.0554
	Avg/Unmap	2.6950	2.9746	2.4404	2.1645
	Avg/Map7	1.2478	1.4304	1.1488	0.8219
Target Ghosts	CAMD/Unmap	37.14	42.97	31.25	17.26
	CAMD/Map7	26.43	20.31	13.94	-2.98
	GAMD/Unmap	80.00	76.56	79.81	75.60
	GAMD/Map7	81.07	70.31	78.37	76.79
Avoidance Ghosts	CMD/Unmap	20.00	43.75	23.08	28.57
	CMD/Map7	54.29	75.00	53.85	66.67

Table 6.14: Characterization of the Cyclin Core Region Motif Data Set

Motif Characteristic		Motifs	
		A	B
Motif Length		23	18
Relative Entropy	Ovr/Unmap	0.2054	0.2123
	Ovr/Map7	0.0921	0.0433
	Avg/Unmap	2.2682	2.2473
	Avg/Map7	1.3076	1.0422
Target Ghosts	CAMD/Unmap	26.81	26.85
	CAMD/Map7	18.12	9.26
	GAMD/Unmap	77.54	75.00
	GAMD/Map7	71.74	74.07
Avoidance Ghosts	CMD/Unmap	26.09	33.33
	CMD/Map7	56.52	61.11

Table 6.15: Characterization of the Acetyltransferase Motif Data Set. See main text for explanation of terms.

Motif Characteristic		Motif Detection Mode	
		Unmap	Map7
Motif Length		0.5061 ****	0.5818 *****
Relative Entropy	Ovr/Unmap	-0.4231 **	-0.3301 *
	Ovr/Map7	-0.3770 **	-0.1378
	Avg/Unmap	0.4581 ***	0.1168
	Avg/Map7	0.1466	0.1764 *
Target Ghosts	CAMD/Unmap	0.6642 ******	0.3769 **
	CAMD/Map7	0.5344 ****	0.3955 **
	GAMD/Unmap	0.5301 ****	0.5187 ****
	GAMD/Map7	0.4971 ***	0.4146 **
Avoidance Ghosts	CMD/Unmap	-0.3128 *	0.0181
	CMD/Map7	-0.0482	0.0650

(a)

Motif Characteristic		Motif Detection Mode	
		Unmap	Map7
Motif Length		0.4166 *	0.5210 **
Relative Entropy	Ovr/Unmap	-0.5047 **	-0.2719 *
	Ovr/Map7	-0.4673 **	-0.0539
	Avg/Unmap	0.1366	-0.1625
	Avg/Map7	-0.2013	0.0492
Target Ghosts	CAMD/Unmap	0.4646 **	0.0810
	CAMD/Map7	0.0622	0.0675
	GAMD/Unmap	0.6307 ****	0.6096 ****
	GAMD/Map7	0.5196 **	0.3304 *
Avoidance Ghosts	CMD/Unmap	0.0873	0.4061 *
	CMD/Map7	0.0371	0.1570

(b)

Table 6.16: Real Data Results: Motif Detection / Characteristic Correlations. The probability α that each correlation coefficient is not significantly different from a hypothesis of no correlation was calculated using the t-test described on pages 250 and 251 of [MMH90]. These probabilities are represented by starred annotation as follows: **** $\rightarrow \alpha = 0.05$, *** $\rightarrow \alpha = 0.1$, ** $\rightarrow \alpha = 0.2$, and * $\rightarrow \alpha = 0.5$. See main text for explanation of terms. a). All motifs; b) Acetyltransferase B, Cyclin IV, Kinase I / VI / VII Motifs Omitted.

suggested that motif conservation and the distinctness of the overall symbol-distribution of a motif from the sequence background were important factors; however, with the exception of a moderately strong correlation of unmapped mode detect ability with the average unmapped relative entropy, these factors do not seem to play a significant role. Indeed, it seems that the most important factors are motif length and various measures of the closeness of target sequence ghosts (in the case of unmapped mode) and motif length and the closeness of target sequence ghosts to the known motifs as opposed to the consensus strings (in the case of Map7 mode). As the strongest correlations stemmed from target sequence ghosts and the interference caused by these ghosts may be masking the effects of other factors, we removed the data for the five cases with the closest target ghosts (Acetyltransferase B, Cyclin IV, Protein Kinase I, VI, and VII) and recomputed the correlation coefficients. These results, given in Table 6.16,(b) do reduce the significance of many previous correlations, though correlations with target sequence ghosts remain strong. It is difficult to do further removal of motifs to explore other factors in a dataset this small without rendering the derived correlation coefficients indistinguishable from the effects of sampling error. However, the results we have do suggest that avoidance sequence ghosts play a larger role in Map7

mode once the effect of target sequence ghosts are removed. This is consistent with our experience, which has shown that there are typically many more ghosts in avoidance sequences that are much closer to the threshold under Map7 mode than in unmapped mode. This may be a product of the non-random manner in which amino acids occur in actual proteins, and the manner in which Map7 mode "blurs" patterns under this non-random distribution; this is, however, to be expected, as it is known that protein sequences in general and motifs in particular may contain clusters of biochemically similar amino acids.

6.3 Summary

In any case, we seem to have reasonable hypotheses for the discrepancy of the performance of our algorithm on simulated and real datasets:

1. Unlike our simulated datasets, real datasets may incorporate many copies of a motif of varying degrees of conservation.

2. Unlike our simulated datasets constructed in Section 4.5, real datasets have a non-random distribution of amino acids, and this non-randomness is amplified by using the Map7 mapping.

In practice, the former is not a problem; our algorithm will simply find motifs and users will subsequently search sequences to locate any well-conserved ghosts of those motifs. One could even escape this problem altogether by applying the modifications described in this thesis to more complex versions of the Gibbs algorithms that explicitly allow and automatically detect multiple copies of a motif in a sequence [LAB+93, LNL95, NLL95], or by adopting a strategy of erasing motifs as they are found in order to expose the ghosts in subsequent motif searches [BE95]. Ironically, the problem here is our criterion of detection, which is based on the assumption that only one copy of a motif can occur in a sequence. This highlights a (perhaps common) misconception that conserved regions in alignments can be treated as uniquely-occurring motifs. This is, in retrospect, not unexpected, as alignments are concerned with co-linear occurrences of distinct patterns, and do not in themselves flag multiple occurrences of these patterns within sequences; however, users of these alignments should still be careful. The latter difficulty with Map7 mode may possibly be alleviated by modifications that allow the Gibbs algorithms to use similarity-matrix schemes for encoding amino acid similarity; however, the experience reported on page 213 of [LAB+93] suggests otherwise.

Note that some of the most interesting implications of the results in this thesis stem from the discrepancy between the results for complex real datasets whose characteristics are unknown and simple simulated datasets whose characteristics are under our control. In particular, we would not be aware of the degree of sensitivity of our algorithm's performance to target and avoidance ghosts unless this discrepancy had happened and forced us to reexamine our hypotheses about how our algorithm behaves and what motifs really mean. This brings home another important caveat: when examining an algorithm's behavior, simulated datasets allow us to reason systematically about this behavior in a rigorous manner, while real datasets ensure that our reasoning remains relevant. Hence testing relative to both kinds of datasets is neccessary.

Chapter 7

Conclusions and Future Work

The main results are summarized as followings:

1. The simulation study #1 tells us that both of naive algorithm and our algorithm perform well at low motif dispersion rates, and performance falls off as the motif dispersion rate increases (particularly in the case of gapped motifs). Our algorithm always performs better than the naive Gibbs algorithm in the case of ungapped motifs and gapped protein motifs.

2. In simulation study #2, when multiple motifs are present, our algorithm is able to detect more than half of the target motifs after only 10 runs and almost all of the target motifs after 25 runs under all motif dispersion rates in the case of ungapped DNA and ungapped and gapped protein motifs.

3. In general, the modified algorithm performs better on protein than DNA datasets for both ungapped and gapped motifs.

4. Amino acid similarity grouping information does help on detecting target motifs in some cases, for instance, we were only able to detect the target motif A of acetyltransferase proteins in Map7 mode.

5. The modified algorithm successfully detected many target motifs in real datasets. For instances, cytosine methyltransferase target motif-IV, VI and VIII; protein kinase target motif-IV and VII; lipocalin target motif A and B, cyclin protein target motif-I, II and III; acetyltransferase protein target motif A.

Besides, we also notice that the alphabet size of target sequences play a very important role in the performance of our algorithm. Our algorithm seems preferable to larger alphabet size sequences like proteins. The reason for this is still unknown. There could be two possibilities. First, our algorithm inherited this fact from the modified Gibbs sampling algorithm [RT98]. Second, the parameters used in the tests were not optimized. For example of pseuodocounts maybe need to be adjusted in term of different alphabet size sequences.

Regarding the future work, several things closely related to the algorithm performance improvements and extensions:

1. Perform parametric study on pseudocount, size of sequence alphabet, gap penalties, motif length, threshold value and convergence parameter. As suggested in [Gus97, page 312-321], we can partition the parameter space into regions such that in each region one alignment is optimal throughout and such that each region is maximal for this property. Thus the parametric study allows us to see explicitly, and completely, the effect of parameter choices on the optimal alignment.

2. Explore different weighting scheme on avoidance sequences. There are other possible ways in which avoidance sequence information can be incorporated into the Gibbs sampling algorithm to influence motif model. At the place of the creation of the weights for candidate motif-instances in the selected sequence s, instead of using the method described in Section 2.3, we may modify these weights (to use the original terminology of [LAB+93]) is to replace weight Q_x/P_x by $(Q_x/P_x)+(Q_x/B_{avoidance})$, where $B_{avoidance}$ is the probability associated with the best match of the motif model for the target sequences against any substring of any sequence in the set of avoidance sequences. This is intuitively appealing, as it implies that selected models optimize the distinctness of the motif model from both the background symbol distribution in the target sequences *and* the set of all possible motif-instances in the avoidance

sequences. As the magnitudes of the individual ratios are not known, it may be best to normalize both ratios over their maximum possible values and take either a linear combination, i.e.,

$$c_1 \frac{Q_x}{P_x} + c_2 \frac{Q_x}{B_{avoidance}}$$

for constants c_1 and c_2, or a more exotic function, e.g.,

$$c_1 \frac{Q_x}{P_x} + c_2 e^{\frac{Q_x}{B_{avoidance}}}$$

of these normalized ratios to more accurately reflect the relative importance of each term. This may ultimately be an application-specific matter. For instance, some applications may stress the quality of the match to the target sequences while others may require maximal distinctness from the avoidance sequences.

3. Extend our modified algorithm to seek several target motifs simultaneously rather than sequentially, which allows information gained about one to aid the alignment of the others (see [LAB+93], [LL95] and [LNL95] for detail). This might help to solve the "ghost" problem in real dataset tests described in Chapter 6.

4. The general stochastic algorithm for motif identification given in Section 2.3 can be adapted to describe other stochastic heuristic algorithms for finding motifs such as those using Hidden Markov Models (HMM)

[DEKM98]. One of our project member (Mr. Chris Trendall) has applied the modifications sketched in Section 2.3 to implement an HMM algorithm for target motif identification, and preliminary tests relative to ungapped motif DNA sequence datasets have shown that this algorithm has performance comparable to the modified Gibbs algorithm described in this thesis. We should also test the modified HMM algorithm on other simulated and real datasets and compare these results with those reported in this thesis.

Bibliography

[BE95] Timothy L. Bailey and Charles Elkan. 1995. Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization. *Machine Learning Journal*, 21, 51–83.

[CNJ90] S. W. Cowan, M. E. Newcomer, T. A. Jones, 1990, Crystallographic refinement of human serum retinol binding protein at 2A resolution. *Proteins* 8(1):44-61.

[DEKM98] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

[GJ79] Michael R. Garey and David S. Johnson. 1979.

[Gus97] Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press.

[HQH88] Steven K. Hanks, Anne Marie Quinn, and Tony Hunter. 1988. The Protein Kinase Family: Conserved Features and Deduced Phylogeny of the Catalytic Domains. *Science*, 241, 42–52.

[HT91] Hunter, T. 1991. Protein kinase classification. *Methods in Enzymology*, 200, 3–37.

[ISNH94] Minoru Ito, Kumiyasu Shimizu, Michio Nakanishi, and Akihiro Hashimoto. 1994. Polynomial-Time Algorithms for Computing Characteristic Strings. In Maxime Crochemore and Dan Gusfield, eds., *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM'94)*, pages 289–306. Lecture Notes in Computer Science no. 807. Springer-Verlag, Berlin. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco.

[JTWZ99] Tao Jiang, Chris G. Trendall, H. Todd Wareham, and Xian Zhang 1999. Stochastic Heuristic Algorithms for Target Motif Identification. *RECOMB*, poster.

[JTW99] Tao Jiang, Chris G. Trendall, H. Todd Wareham, and Xian Zhang 1999. Stochastic Heuristic Algorithms for Target Motif Identification. *submitted to Journal of Computation Biology.*

[LAB+93] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wooton. 1993. Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment. *Science*, 262, 208–214.

[LL+99] J. Kevin Lanctot, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. 1999. Distinguishing String Selection Problems. In *ACM-SIAM SODA '99.*

[LL95] Jun S. Liu and Charles E. Lawrence. 1995. Statistical Models for Multiple Sequence Alignment: Unifications and Generalizations. In *American Statistical Association: 1995 Proceedings of the Statistical Computing Section*, pages 1–8.

[LNL95] Jun S. Liu, Andrew F. Neuwald, and Charles E. Lawrence. 1995. Bayesian Models for Multiple Local Sequence Alignment and Gibbs Sampling Strategies. *Journal of the American Statistical Association*, 90(432), 1156–1170.

[LS87] L. Sawyer. 1987. Protein structure. One fold among many. *Nature* Jun 25-Jul 1;327(6124):659.

[MMH90] Richard B. May, Michael E.J. Masson, and Michael A. Hunter. 1990. *Applications of Statistics in Behavioral Research.* Harper and

Row, New York.

[MVF94] Marcella A. McClure, T. k. Vasi and W. M. Fitch. 1994. Comparative Analysis of Multiple Protein-Sequence Alignment Methods. *Mol. Biol. Evol.*, 11(4), 571–592.

[NG94] Andrew F. Neuwald and Philip Green. 1994. Detecting Patterns in Protein Sequences. *J. Mol. Biol.*, 239, 698–712.

[NLL95] Andrew F. Neuwald, Jun S. Liu, and Charles E. Lawrence. 1995. Gibbs motif sampling: Detection of bacterial outer membrane protein repeats. *Protein Science*, 4, 1618–1632.

[OL91] P. O'Farrell and P. Léopold. 1991. A Consensus of Cyclin Sequences Reveals Homology with the *ras* Oncogene. *Cold Spring Harbor Symposium on Qualitative Biology*, LVI, 83–92.

[PBR89] Janos Posfai A. S. Bhagwat G. Posfai J. Roberts. 1989. Predictive motifs derived from cytosine methyltransferases. *Nucleic Acids Research*, Vol. 7, 2421-2435.

[RT98] Emily Rocke and Martin Tompa. 1998. An Algorithm for Finding Novel Gapped Motifs in DNA Sequences. In Sorin Istrail, Pavel Pevzner, and Michael Waterman, eds., *Proceedings of The Second*

Annual International Conference on Computational Molecular Biology (RECOMB'98), pages 228–233. ACM Press, New York.

[Tom98] Martin Tompa. 1999. Personal communication.

[Tom99] Martin Tompa. 1999. Personal communication.

[W95] Michael S. Waterman, 1995, Introduction to computational biology, *Chapman & Hall Press.*

[Wan98] Joe Wang. 1998. Personal communication.