Predicting Location and Structure of Beta-Sheet Regions Using Stochastic Tree Grammars

Hiroshi Mamitsuka and **Naoki Abe**
Theory NEC Laboratory, RWCP*
c/o NEC C & C Research Laboratories, 4-1-1 Miyazaki Miyamae-ku, Kawasaki,216 Japan.
Tel. 81-44-856-2143, Fax. 81-44-856-2235.
E-mail : {mami, abe}@sbl.cl.nec.co.jp.

ABSTRACT

We describe and demonstrate the effectiveness of a method of predicting protein secondary structures, β-sheet regions in particular, using a class of stochastic tree grammars as representational language for their amino acid sequence patterns. The family of stochastic tree grammars we use, the Stochastic Ranked Node Rewriting Grammars (SRNRG), is one of the rare families of stochastic grammars that are expressive enough to capture the kind of long-distance dependencies exhibited by the sequences of β-sheet regions, and at the same time enjoy relatively efficient processing. We applied our method on real data obtained from the HSSP database and the results obtained are encouraging: Using an SRNRG trained by data of a particular protein, our method was actually able to predict the location and structure of β-sheet regions in a number of different proteins, whose sequences are less than 25 per cent homologous to the training sequences. The learning algorithm we use is an extension of the 'Inside-Outside' algorithm for stochastic context free grammars, but with a number of significant modifications. First, we restricted the grammars used to be members of the 'linear' subclass of SRNRG, and devised simpler and faster algorithms for this subclass. Secondly, we reduced the alphabet size (i.e. the number of amino acids) by clustering them using their physicochemical properties, gradually through the iterations of the learning algorithm. Finally, we parallelized our parsing algorithm to run on a highly parallel computer, a 32-processor CM-5, and were able to obtain a nearly linear speed-up. We emphasize that our prediction method already goes beyond what is possible by the homology-based approaches. We also stress that our method can predict the *structure* as well as the location of β-sheet regions, which was not possible by previous inverse protein folding methods.

*Real World Computing Partnership.

Keywords: Protein secondary structure prediction, Beta-sheets, Stochastic tree grammars, Maximum likelihood estimation, Minimum description length.

Introduction

The problem of predicting protein structures from their amino acid sequences is probably the single most important problem in genetic information processing with immense scientific significance and broad engineering applications. The *secondary* structure prediction problem, namely the problem of determining which regions in a given amino acid sequence correspond to each of the three categories, α-helix, β-sheet, and others, is considered to be an important step towards this goal, and has been attempted by many researchers (e.g. (Rost & Sander 1993)). No method to date, however, has achieved a prediction accuracy much higher than 70 per cent, casting serious doubt as to whether a significantly better performance is achievable by any approach along this line.

Motivated largely by the apparent limitation of residue-wise secondary structure prediction methods, more 'knowledge intensive' approaches to the problem of protein structure prediction have been proposed and investigated, including the homology-based approach (Blundell *et al.* 1987) and the 'inverse protein folding' approach (Bowie, Luthy, & Eisenberg 1991). More recently it has been proposed (Chothia 1992) that all protein structures (foldings) found in today's living organisms can be classified into a relatively small number (less than a thousand or so) of types, in confirmation of such knowledge intensive approaches. In a knowledge-based approach, the prediction method keeps effectively a catalogue of patterns of amino acid sequences corresponding to existing types of protein structures, and prediction on a new sequence is done by simply finding those patterns that match parts of the input sequence. The central issue here then is how to represent these patterns with a sufficient and appropriate level of generalization. Abe and Mamitsuka have re-

cently proposed to use a certain class of stochastic tree grammars called the Stochastic Ranked Node Rewriting Grammars (SRNRG) as representational scheme for sequence patterns of protein secondary structures, especially those of β-sheets (Abe & Mamitsuka 1994). The primary goal of the present paper is to demonstrate its effectiveness by further experimental results.

The problem of predicting β-sheet regions has been considered difficult because β-sheets typically range over several discontinuous sections in an amino acid sequence, and their sequences exhibit long distance dependency. The family of stochastic tree grammars we use in the present paper (SRNRG) is suitable for expressing the kind of long-distance dependencies exhibited by the sequences of β-sheet regions, such as the 'parallel' and 'anti-parallel' dependencies and their combinations. RNRG was originally introduced in the context of computationally efficient learnability of grammars in (Abe 1988), and its discovery was inspired by the pioneering work of Joshi et al. (Joshi, Levy, & Takahashi 1975; Vijay-Shanker & Joshi 1985) on a formalism for natural language called 'Tree Adjoining Grammars' (TAG). SRNRG is one of the rare families of grammatical systems that have both enough expressive power to cope with such dependencies and at the same time enjoy efficient parsability and learnability.[1] In particular, SRNRG has expressive power exceeding those of both Hidden Markov Models (HMMs) and stochastic context free grammars (SCFGs), and yet allows the existence of polynomial time parsing and local optimization algorithm for the maximum likelihood settings of probability parameters.[2]

We designed and implemented a method for predicting β-sheet regions using SRNRG as the representational language. Our prediction method receives as input amino acid sequences with the location of β-sheet regions marked, and trains the probability parameters of an SRNRG, so that its distribution best approximates the patterns of the input sample. Some of the rules in the grammar are intended *a priori* for generating β-sheet regions and others for non-β-sheets. After training, the method is given a sequence of amino acids with *unknown* secondary structure, and predicts according to which regions are generated by the β-sheet rules, in the *most likely* parse for the input sequence.

The learning algorithm we use is an extension of the 'Inside-Outside' algorithm for the stochastic context free grammars.[3] In order to reduce the rather high computational requirement[4] of the learning and parsing algorithms, we have restricted the form of grammars to a certain subclass of RNRG which we call the 'linear RNRG,' and devised a simpler and faster learning algorithm for the subclass. We also employed a method of reducing the alphabet size[5] (i.e. the number of amino acids) by clustering them using MDL(Minimum Description Length) approximation and their physico-chemical properties, gradually through the iterations of the learning algorithm.[6] Finally, we also parallelized our parsing algorithm to run on CM-5, a 32-processor parallel machine. We were able to obtain a nearly linear speed-up, and were able to predict the structure of a test sequence of length 50 in about 6 minutes, whereas the same task took our sequential algorithm almost 3 hours.

We applied our method on real data obtained from the HSSP (Homology-derived Secondary Structures of Proteins Ver 1.0 (Sander & Schneider 1991)) database. The results obtained indicate that our method is able to capture and generalize the type of long-distance dependencies that characterize β-sheets. Using an SRNRG trained by data for a particular protein, our method was actually able to predict the location and structure of β-sheets in test sequences of a number of different proteins, which have similar structures but have less than 25 per cent pairwise homology to the training sequences. We also conducted a similar experiment using two proteins, which have quite different structures but share some similar partial structures, having almost no pairwise homology at all. Our method was able to predict two of the three β-strands in the test sequence correctly in this case. We emphasize that, unlike previous secondary structure prediction methods, our method is able to predict the *structure* of the β-sheet, namely the locations of the hydrogen bonds. We also stress that, since the training sequences and the test sequences are less than 25 per cent homologous, all of our prediction problems be-

[1] Searls claimed that the language of β-sheets is beyond context free and suggested that they are indexed languages (Searls 1993). Indexed languages are not recognizable in polynomial time, however, and hence indexed grammars are not useful for our purpose. RNRG falls *between* them and appears to be just what we need.

[2] Both HMM and SCFG have recently been used in the context of genetic information processing (Brown *et al.* 1993; Asai, Hayamizu, & Onizuka 1993; Sakakibara *et al.* 1994).

[3] It is also related to the extension of the Inside-Outside algorithm developed for the stochastic tree adjoining grammars (Schabes 1992).

[4] The time complexity of the Inside-Outside algorithm for RNRG of a bounded 'rank' k is roughly $O(n^{3(k+1)})$.

[5] As is well known, there are twenty amino acids, and hence we are dealing with an alphabet of size 20.

[6] The physico-chemical properties we use are the molecular weight and the hydrophobicity, which were used in (Mamitsuka & Yamanishi 1992) in their method for predicting α-helix regions.

long to what is sometimes referred to in the literature as the 'Twilight Zone' (Doolittle *et al.* 1986), where alignment is no longer effective.

Modeling Beta Sheet Structures with RNRG

We first briefly review the definition of the Ranked Node Rewriting Grammar (RNRG) and give some illustrative examples.[7] An RNRG is a tree generating system, and consists of a single tree structure called the *starting* tree, and a finite collection of rewriting rules which rewrite a node in a tree with an incomplete tree structure. The node to be rewritten needs to be labeled with a *non-terminal* symbol, and must have the same number of descendants (called the 'rank' of the node) as the number of 'empty nodes' in the incomplete tree structure. After rewriting, the descendants of the node are attached to these empty nodes in the same order as before rewriting. The string language of the grammar is the set of *yields* of the trees generated by the grammar, namely the strings that appear on the leaves of the trees. If we place an upper bound, say k, on the rank of a node that can be rewritten, we obtain families of grammars, RNRG(k), each of which has varying expressive power. The string languages of RNRG(0), denoted RNRL(0), equal the context free languages (CFL), those of RNRG(1) equal the tree adjoining languages (TAL), and for any $k \geq 2$, RNRL(k) properly contains RNRL($k-1$). We now give some examples of RNRG grammars. The language $L_1 = \{ww^Rww^R | w \in \{a,b\}\}$ is generated by the RNRG(1) grammar G_1 shown[8] in Figure 1(a). The '3 copy' language $L_2 = \{www \mid w \in \{a,b\}^*\}$ can be generated by the RNRG(2) grammar G_2 shown in Figure 1(b). Note that L_1 can be generated by a tree adjoining grammar, but not L_2. The way the derivation in RNRG takes place is illustrated in Figure 2, which shows the derivation of the string 'ababab' by G_2. Each of the trees shown in Figure 2 is called a 'partially derived tree.' Note that the tree structure introduced by a particular rule may be split into several pieces in the final derived tree, unlike usual parse trees in CFG. (In the figure, the part of the derived tree introduced by (α_1) is indicated in a thick line.) Given the definition of RNRG, the *stochastic* RNRG is defined analogously to the way stochastic CFG is defined from CFG. That is, associated with each rewriting rule in a stochastic RNRG is its *rule application probability*, which is constrained so that for each non-terminal, the sum total of rule application probabilities of all the rewriting rules for that non-terminal equals unity. This way, each stochastic RNRG can be viewed as a probabilistic generator of finite strings, and defines a probability distribution over the set of all finite strings.

Next some typical β-sheet structures are illustrated in Figure 3. Figure 3(e) shows a picture of an actual β-sheet structure (Branden & Tooze 1991), whereas all others are schematical representations. The arrows indicate the β-sheet strands, and the line going through them the amino acid sequence. The β-sheet structure is retained by hydrogen bonds (H-bonds) between the corresponding amino acids in neighboring strands, so it is reasonable to suspect that there are correlations between the amino acids in those positions. The structure exhibited in Figure 3 (a) is known as the 'anti-parallel' β-sheet, as the dependency follows the pattern *abc..cba..abc...cba*, where the use of a same letter indicates that those positions are connected by H-bonds and believed to be correlated. In contrast, the structure exhibited in Figure 3 (b) is known as the 'parallel' β-sheet, since the dependency here is of the pattern *abc..abc...* Both of these types of dependency can be captured by RNRG, in particular, G_1 and G_2 in Figure 1, respectively. These structures can be combined to obtain larger β-sheets, as is shown in Figure 3(d) and can result in a high degree of complexity, but they can be handled by an RNRG of a higher rank.

Learning and Parsing of The Linear Subclass

The 'linear' subclass of RNRG we use in this paper is the subclass satisfying the following two constraints: (i) Each rewriting rule contains at most one node labeled with a non-terminal symbol of rank greater than 0; (ii) Every other non-terminal (of rank 0) is a 'lexical' non-terminal, namely all rewriting rules for it are of the form $A \rightarrow a$ for some terminal symbol a. Examples of RNRG of rank 1 satisfying these constraints can be found, for example, in Figure 4(a). Note that each occurrence of a lexical non-terminal can be thought of as defining a distribution over the alphabet, and this is written in as part of the rule in the figure. In our current scenario in which the alphabet equals the amino acids, the rule application probabilities for these lexical rules are called 'amino acid generation probabilities.' With these constraints, the parsing and learning algorithms can be significantly simplified.

The Learning Algorithm

Our learning algorithm is an extension of the 'Inside-Outside' algorithm for SCFG (Jelinik, Lafferty, & Mercer 1990) and it is a local optimization algorithm for

[7] We refer the interested reader to (Abe 1988) for the detailed definition.

[8] Note that 'λ' indicates the empty string, and an edge leading to no letter leads to an empty node.

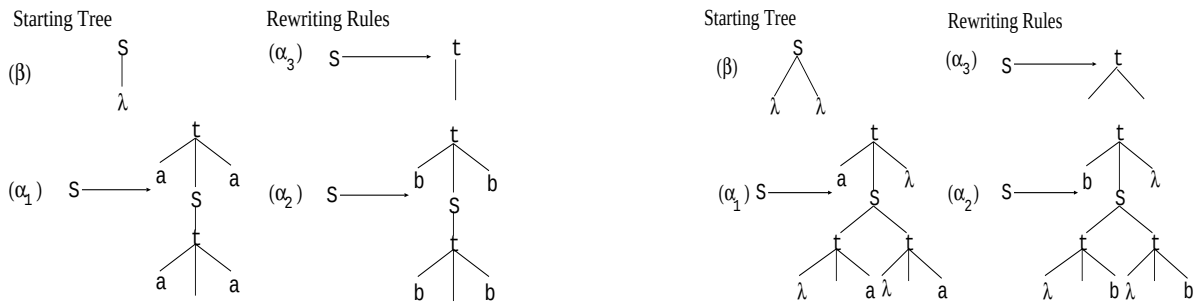


Figure 1: (a) RNRG(1) grammar G_1 and (b) RNRG(2) grammar G_2.

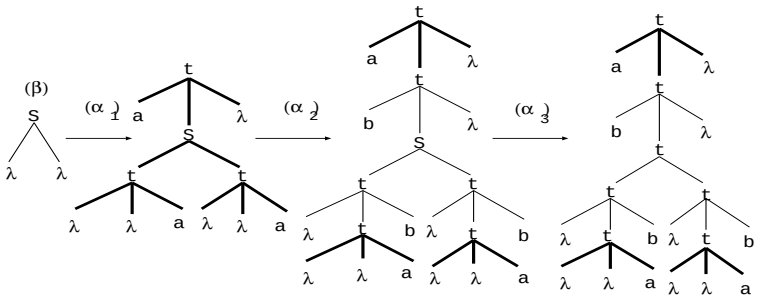


Figure 2: Derivation of 'ababab' by an RNRG-2 grammar

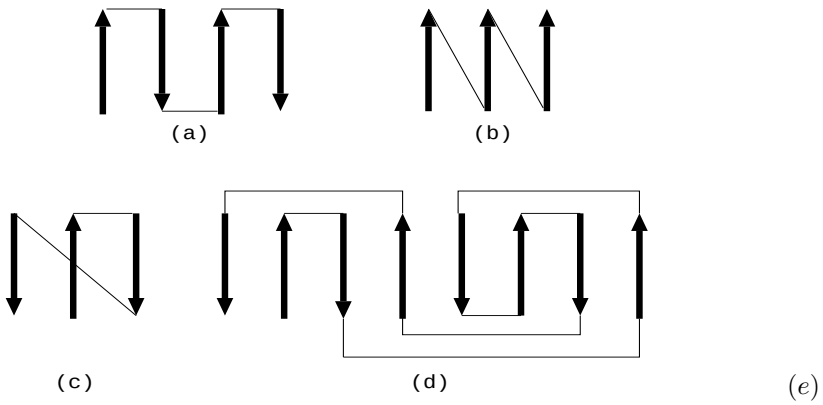


Figure 3: Some typical β-sheet structures

the maximum likelihood settings of the rule application probabilities and the amino acid generation probabilities in the input grammar. The algorithm is an iterative procedure which re-estimates and updates its current settings of all of its probability parameters. The re-estimation of the rule application probabilities is done essentially by taking the weighted frequency of each rule application, weighted according to the likelihood assigned to the derivations by the current parameter settings, divided by the weighted frequency of the non-terminal on the right hand side of that rule. These weighted frequencies can be calculated in terms of what are called the 'inside' and 'outside' probabilities, which correspond to the 'forward' and 'backward' probabilities in the Baum-Welch algorithm for HMM (Rabiner & Juang 1986). The re-estimation procedure is guaranteed to increase the likelihood assigned to the sample, as is the case with the Baum-Welch re-estimation for HMM. For simplicity, we describe our algorithm for rank 1 linear SRNRG.

Let Ξ be the input sample, and let σ denote an arbitrary sequence in Ξ and N be its length. Let σ_t denote the t-th letter (amino acid) of σ. We define the 'inside probability' of non-terminal S at i, j, k, l on σ, written $In_\sigma[S, i, j, k, l]$, to be the sum total of the probabilities of all partially derived trees whose (two discontinuous) yields match the sub-strings from the i-th to the j-th and from the k-th to the l-th letters of σ. Similarly, we define the 'outside probability' of non-terminal S at i, j, k, l on σ, written $Out_\sigma[S, i, j, k, l]$, to be the sum total of the probabilities of all partially derived trees whose (three discontinuous) yields match the sub-strings from the first to the i-th, from the j-th to the k-th, and from the l-th to the N-th letters of σ.

Now let G be the input grammar, and let $N(G)$ denote the set of non-terminals of G. For each rewriting rule r in G, let $T(r)$ denote the rule application probability of r, and $L(r)$ and $R(r)$ the non-terminal of the left hand side and the (unique) non-terminal on the right hand side of r, respectively. Let n^r_f, $f = 1, ..., 4$, denote the number of terminal symbols at each of the four corners (left up, left low, right low, and right up) of the unique (if any) non-terminal symbol in the right hand side of r, and $P^{r,x}_f(\alpha)$, $\alpha = 1, ..., 20$, denote the generation probability of amino acid α at the x-th position in the f corner.

The inside and outside probabilities are calculated as follows. The inside probabilities at arbitrary index (i, j, k, l) can be defined recursively solely in terms of the inside probabilities of 'smaller' intervals, so it can be calculated as long as the inside probabilities at those (i', j', k', l') such that $(j'-i')+(l'-k') < (j-i)+(l-k)$ have already been calculated. The looping used in the procedure for calculating inside probabilities exhibited below ensures that this condition is satisfied.

For $i := N$ to 1
 For $j := i$ to N
  For $k := N$ to $j+1$
   For $l := k$ to N
    For $S \in N(G)$

$$In_\sigma[S,i,j,k,l] = \sum_{\{r \in G | L(r)=S\}} \{\ T(r) \cdot In_\sigma[R(r), i+n^r_1, j-n^r_2, k+n^r_3, l-n^r_4] \cdot \prod_{x=1}^{n^r_1} P^{r,x}_1(\sigma_{i+x}) \prod_{x=1}^{n^r_2} P^{r,x}_2(\sigma_{j-x}) \prod_{x=1}^{n^r_3} P^{r,x}_3(\sigma_{k+x}) \prod_{x=1}^{n^r_4} P^{r,x}_4(\sigma_{l-x})\ \}$$

The outside probabilities can be calculated in a similar fashion, as shown below.

For $i := 1$ to N
 For $j := N$ to $i+1$
  For $k := j$ to N
   For $l := N$ to $k+1$
    For $S \in N(G)$

$$Out_\sigma[S,i,j,k,l] = \sum_{\{r \in G | R(r)=S\}} \{\ T(r) \cdot Out_\sigma[L(r), i-n^r_1, j+n^r_2, k-n^r_3, l+n^r_4] \cdot \prod_{x=1}^{n^r_1} P^{r,x}_1(\sigma_{i-x}) \prod_{x=1}^{n^r_2} P^{r,x}_2(\sigma_{j+x}) \prod_{x=1}^{n^r_3} P^{r,x}_3(\sigma_{k-x}) \prod_{x=1}^{n^r_4} P^{r,x}_4(\sigma_{l+x})\ \}$$

Given the inside and outside probabilities, the updates of the rule application probabilities, as well as the amino acid generation probabilities, can be calculated as follows. Let $U_\sigma(r)$ denote the 'weighted average frequency' of rewriting rule r, namely the average frequency of r in a single parse of σ, weighted according to the generation probability of each parse using the current settings of the probability parameters. Similarly define $V^{r,f,x}_\sigma(\alpha)$ to be the weighted average frequency of amino acid α at the x-th position of corner f in rule r in a single parse of σ. Now if we let $Pr_\sigma[r, i, j, k, l]$ denote the probability that the grammar generates the input sequence σ *and* uses rule r at the (i, j, k, l) position, then it can be calculated as follows.

$$Pr_\sigma[r,i,j,k,l] = Out_\sigma[L(r), i, j, k, l] \cdot In_\sigma[R(r), i+n^r_1+1, j-n^r_2-1, k+n^r_3+1, l-n^r_4-1] \cdot$$

$$T(r) \cdot \prod_{x=1}^{n_1^r} P_1^{r,x}(\sigma_{i+x}) \prod_{x=1}^{n_2^r} P_2^{r,x}(\sigma_{j-x})$$
$$\prod_{x=1}^{n_3^r} P_3^{r,x}(\sigma_{k+x}) \prod_{x=1}^{n_4^r} P_4^{r,x}(\sigma_{l-x})$$

Now $U_\sigma(r)$ can be calculated by summing $Pr_\sigma[r,i,j,k,l]$ over all i,j,k,l, and dividing it by the likelihood assigned to σ by the current parameter settings, $P(\sigma)$.

$$U_\sigma(r) = \frac{\sum_i \sum_j \sum_k \sum_l Pr_\sigma[r,i,j,k,l]}{P(\sigma)}$$

Similarly $V_\sigma^{r,f,x}(\alpha)$ can be calculated as follows. (We show the case $f = 1$ only.)

$$V_\sigma^{r,1,x}(\alpha) = \frac{\sum_i \sum_j \sum_k \sum_l \sum_{\sigma_{i+x}=\alpha} Pr_\sigma[r,i,j,k,l]}{P(\sigma)}$$

Finally, from $U_\sigma(r)$ and $V_\sigma^{r,f,x}(\alpha)$, the update values for the rule application probabilities $T(r)$ and the amino acid generation probabilities $P_f^{r,x}(\alpha)$ can be calculated as follows.

$$T(r) = \frac{\sum_{\sigma\in\Xi} U_\sigma(r)}{\sum_{\sigma\in\Xi} \sum_{\{r'\in G | L(r')=L(r)\}} U_\sigma(r')}$$

$$P_f^{r,x}(\alpha) = \frac{\sum_{\sigma\in\Xi} V_\sigma^{r,f,x}(\alpha)}{\sum_{\sigma\in\Xi} \sum_{\alpha'} V_\sigma^{r,f,x}(\alpha')}$$

The above process is repeated until some stopping condition is satisfied, usually till the changes in the probability parameters become smaller than a certain preset amount.

The parsing algorithm can be obtained by replacing '$\sum$' by 'max' in the definition of the 'Inside' algorithm, and retaining the most likely sub-parse at any intermediate step.

Reducing the Alphabet Size with MDL Approximation

After each iteration of the above learning algorithm at each lexical rule, we attempt to merge some of the amino acids, if the merge reduces the total description length (approximated using the probability parameters calculated up to that point). For this purpose we make use of the Euclidean distance between the 20 amino acids in the (normalized) 2-dimensional space defined by their molecular weight and hydrophobicity. At each iteration, we select the two among the clusters from the previous iteration, which are *closest* to each other in the above Euclidean space, and merge them to obtain a single new cluster, *provided* that the merge results in reducing the following approximation of 'description length,' where we let $c \in C$ be the clusters, $P(c)$ the sum total of generation probabilities of amino acids in the cluster c, and m the *effective* sample size, namely the weighted frequency of the lexical rule in question in the parses of the input sample, i.e. $m = \sum_{\sigma\in\Xi} \sum_\alpha V_\sigma^{r,f,x}(\alpha)$.

$$-\sum_{c\in C} P(c) \log \frac{P(c)}{|c|} + \frac{|C| \log m}{2}.$$

Note that the above approximation of description length by the average minus log-likelihood of the current values of the probability parameters is accurate only if those probability values are reliable. The algorithm keeps merging more clusters in this fashion, but once it fails to merge one pair, it will not try to merge any other pair in the same iteration, in order to ensure that the merge process does not take place too fast.

Parallel Implementation of Parsing Algorithm on CM-5

As we noted in Introduction, we parallelized our parsing algorithm to run on a 32-processor CM-5. In parallelizing this algorithm, we isolated the data dependency by introducing as the outmost loop parameter, d ($d = (j-i)+(l-k)$ in the rank 1 case), which stands for the total length of all the sub-strings that are *outside* those designated by the current indices. That is, we replace the first four **For** loops in the algorithm for calculating the outside probabilities by those shown below.

For $d := N$ to 1
  For $i := 0$ to $N - d$
    For $j := i + 1$ to $i + d$
      For $l := i + d$ to N

This way, the computation of all table entries for a given d could in principle be performed in parallel. In order to keep our parallel algorithm simple, we designated one of the 32 processors, say P_{31}, as the data routing center to which the other 31 processors send all the updated entries, for each value of d. Every time d is updated, the relevant part of the updated table at P_{31} is copied to the local memory of each of the other 31 processors. As this copying need be done only N times (where N is the length of the input string), the communication overhead occupies an asymptotically diminishing portion of the entire computation time, thus making it possible for us to obtain a nearly linear speed-up, when N is sufficiently large.

Experimental Results

We applied our method on real data obtained from the HSSP database. In our first experiment, we

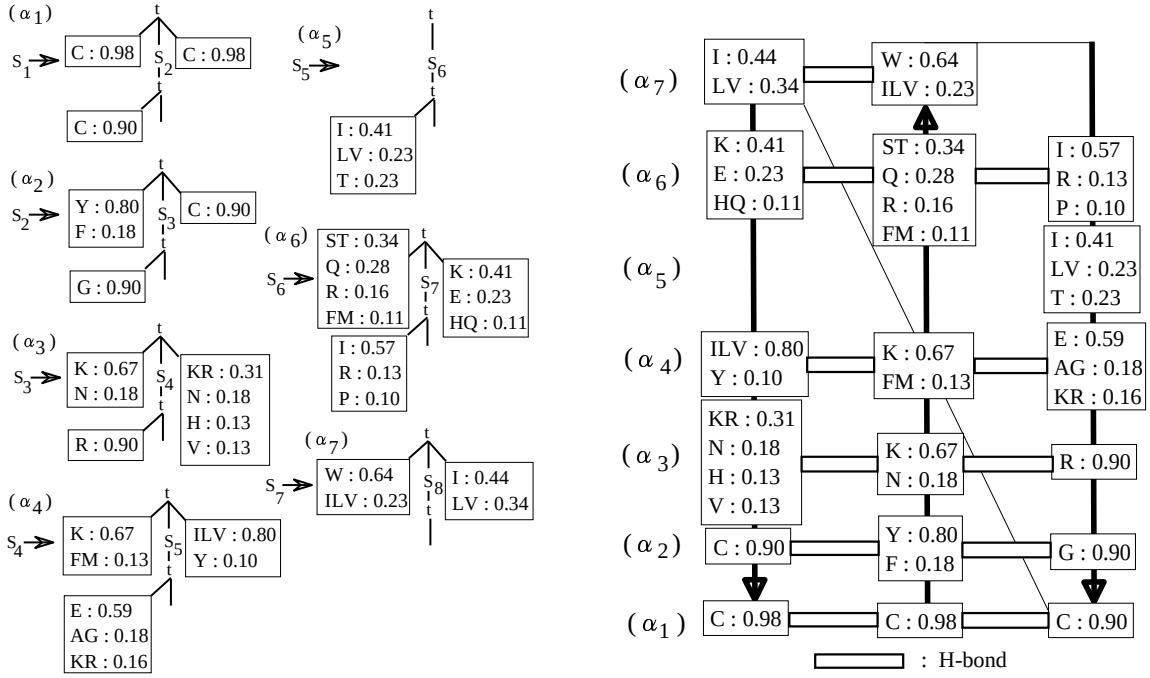


Figure 4: (a) A part of the acquired RNRG grammar and (b) its interpretation.

picked three different proteins, 'Fasciculin' (1fas), 'Caldiotoxin' (1cdta) and 'Neurotoxin B' (1nxb), all of which are toxins. Although these three proteins do have relatively similar structures (Their common structure was shown in Figure 3(e)), their sequences are less than 25 per cent homologous to one another,[9] and hence alignment alone can hardly detect this similarity. We trained a stochastic RNRG with training data consisting effectively of *bracketed* sequences for one of the three proteins, say 1fas, and used the acquired grammar to predict the location of β-sheet regions in an amino acid sequence of another one of the three, either 1cdta or 1nxb. By *bracketing* the input sequences, we mean that we isolated out the (discontinuous) sub-strings of the training sequences that correspond to β-sheets from the rest, and trained the probability parameters of the 'β-sheet rules' in the grammar with them.[10] The probability parameters of the non-β-rules were set to be uniform. We then used the acquired stochastic RNRG grammar to parse an amino acid sequence of either 1cdta or 1nxb, and predicted the location of β-sheet regions according to where the β-sheet rules are in the most likely parse. It was able to predict the location of all three β-strands contained in the test sequence almost exactly (missing only one or two residues which were absent in all of the training data) in both cases. We repeated the same experiment for all (six) possible combinations of the training data and a test sequence from the three proteins. Our method was able to predict all three of the β-strands in all cases, except in predicting the location of β-sheet in a test sequence for 1cdta from training data for 1nxb: It failed to identify one of the three β-strands correctly in this case. The sequences of these toxins were approximately 60 residue long, and the parsing of these sequences required more than an hour on a Silicon Graphics Indigo II graphic workstation.

Figure 4(a) shows the part of the stochastic RNRG(1) grammar obtained by our learning algorithm on the training set for 1fas that generates the β-sheet regions. Note that, in the figure, the amino acid generation probabilities at each position are written in a box. For example, the distribution at the right upper corner in (α_4) gives probability 0.80 to the cluster $\{I, L, V\}$ and probability 0.10 to the single amino acid Y. The interpretation of the grammar is summarized schematically in Figure 4(b). It is easy to see that the grammar represents a class of β-sheets of type (c) in Figure 3. Each of the rules (α_1), (α_2), (α_3), (α_4), (α_6) and (α_7) generates part of the β-sheet region corresponding to a row of H-bonds, and (α_5) inserts an 'extra' amino acid that does not take part in any H-bond. Rule (α_4) says that in the third (from the top)

[9] These were obtained using PDB_SELECT (25 %) developed by Hoboem et. al. (Hoboem *et al.* 1992).

[10] Bracketed input samples are often used in applications of SCFG in speech recognition.

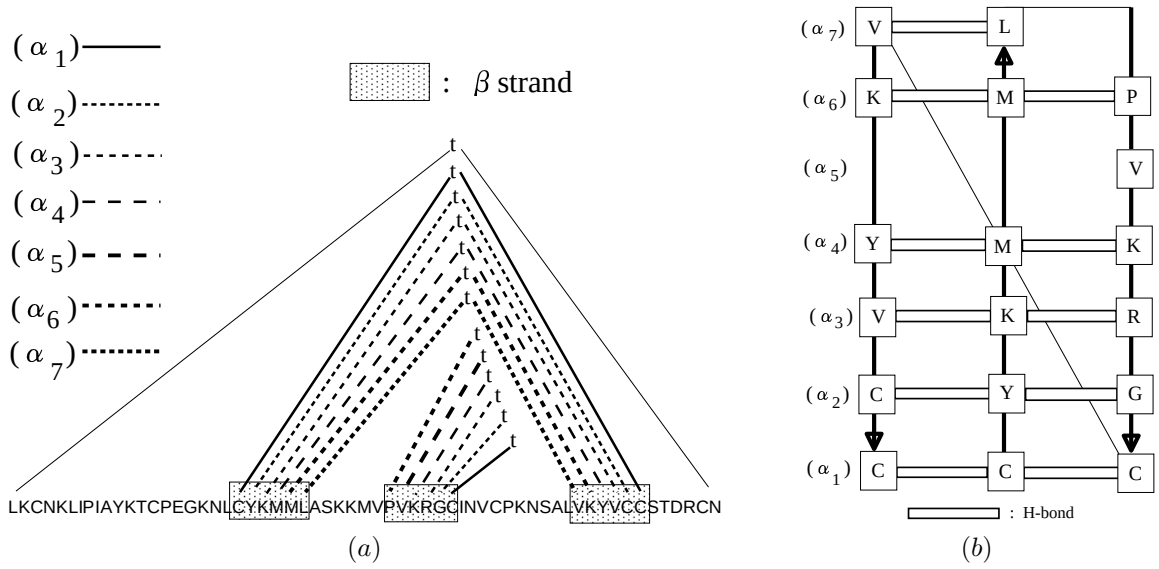


Figure 5: (a) The parse of the test sequence and (b) its interpretation.

row of H-bonds, amino acids I, L and V are equally likely to occur in the leftmost strand, and it is very likely to be K in the middle strand. Note that I, L, and V have similar physico-chemical properties, and it is reasonable that they were merged to form a cluster.

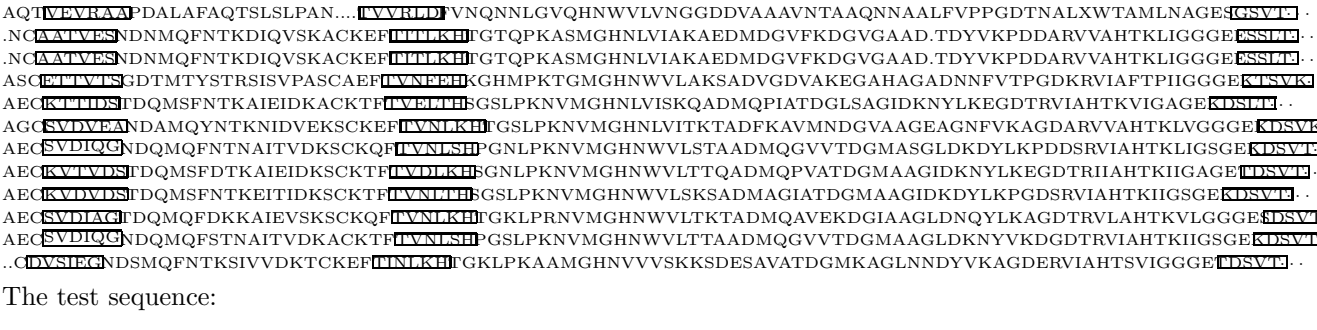
Figure 5(a) shows the most likely parse (derived tree) obtained by the grammar on a test sequence of 1cdta. The shaded areas indicate the actual β-sheet regions, which are *all* correctly predicted. The seven types of thick lines correspond to the parts of the derived tree generated by the seven rules shown in Figure 4(a), respectively. The structural interpretation of this parse is indicated schematically in Figure 5(b), which is also exactly correct. Note that the distributions of amino acids are quite well spread over a large number of amino acids. For example, none of the amino acids in the third strand of the test sequence, except the last two Cs, receives a dominantly high probability in the acquired grammar. The merging of I, L and V mentioned above, therefore, was crucial for the grammar to be able to predict the third strand of the β-sheet in the test sequence.

We also conducted a similar experiment using two proteins, which have very different structures but share some similar partial structures, and have almost no pairwise homology at all. These are 'Azurin' (2azaa) and 'Apoamicyanin' (1aaj), and we trained an SRNRG with sequences of 2azaa and attempted to predict the β-sheet locations in a 1aaj test sequence.

Figure 6 shows the test sequence and part of the training sequences we used. (The actual training sequences are a little longer.) Note that the locations of the β-strands in them are significantly different: The two rightmost β-strands are much closer to each other in the sequence of 1aaj. (Parts of the sequences corresponding to β-strands are indicated with square boxes in the figure.) Our method was able to predict two of the three β-strands (the rightmost two) correctly in this case. The most likely parse is indicated in Figure 7, and the shaded areas indicate the predicted β-sheet regions as before. The reason why the leftmost strand was missed by two residues seems to be that the portion of the test sequence that is predicted to be a β-strand, 'VVDIAK,' is remarkably similar to the actual β-strand portions in a number of training sequences, for example, to 'SVDIAG' of the second sequence to the last. Since the training sequences were obtained by aligning sequences (with unknown secondary structure) to the sequence of 2azaa (with known structure) and marking the regions corresponding to β-sheets in the 2azaa sequence as β-sheets, the reason for this failure might very well be in the process of generating training data by alignment, not in the prediction method itself. The test sequence we used (for '1aaj') was 80 residue long, and the parsing of this sequence took approximately 15 hours on an Indigo II graphic workstation.

Finally, we mention how much speed-up was obtained by our parallel parsing algorithm on CM-5. Figure 8(a) shows the processing time (for parsing) in seconds for input sequences of varying lengths, by our sequential parsing algorithm and by its parallel implementation on CM-5, respectively. Figure 8(b) shows the speed-up factor achieved by the parallel parsing algorithm as a function of the input length. For example, for input length of 50, our parallel implementation

The training data:



The test sequence:

Figure 6: The training data and the test sequence

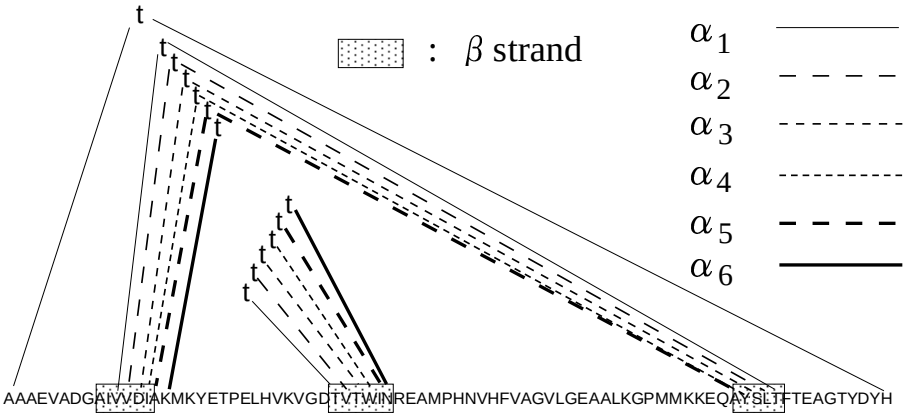


Figure 7: The most likely parse for a test sequence of '1aaj.'

achieved a speed-up factor of 28, a nearly linear speed-up with 31 parallel processors.

Concluding Remarks

We have described a method for predicting protein secondary structures, using a class of stochastic tree grammars as representational language for their amino acid sequence patterns. Our experimental results, admittedly, were only preliminary results in which the test sequences were known to contain relatively similar structures to those of the training sequences. In the future, we hope to demonstrate that our method can achieve a high prediction accuracy on an *arbitrary* test sequence, when equipped with a large catalogue of generalized patterns, expressed as SRNRG rules. One of the most serious obstacles in attempting such a full scale experimentation is the high computational requirement of our method. We hope to overcome this by parallelization and other efforts to speed up our algorithms.

Acknowledgement

We thank Mr. Ken'ichi Takada of NSIS for implementing the parallel parsing algorithm on CM-5, and Mr. Atsuyoshi Nakamura of C & C Research Labs. for his help.

References

Abe, N., and Mamitsuka, H. 1994. A new method for predicting protein secondary structures based on stochastic tree grammars. In *Proceedings of the Eleventh International Conference on Machine Learning.*

Abe, N. 1988. Feasible learnability of formal grammars and the theory of natural language acquisition. In *Proceedings of COLING.*

Asai, K.; Hayamizu, S.; and Onizuka, K. 1993. HMM with protein structure grammar. *Proceedings of the Hawaii International Conference on System Sciences.*

Blundell, T. L.; Sibanda, B. L.; Sternberg, M. J. E.; and Thornton, J. M. 1987. Knowledge-based prediction of protein structures and the design of novel molecules. *Nature* 326:347–352.

Bowie, J. U.; Luthy, R.; and Eisenberg, D. 1991. A method to identify protein sequences that fold into a

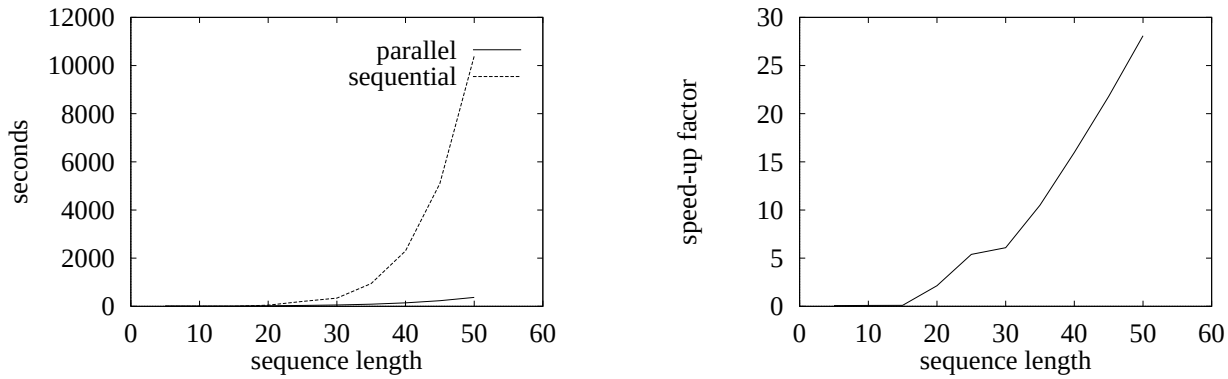


Figure 8: (a) The processing time and (b) the speed-up factor.

known three-dimensional structure. *Science* 253:164–170.

Branden, C., and Tooze, J. 1991. *Introduction to Protein Structure*. New York and London: Garland Publishing, Inc.

Brown, M.; Hughey, R.; Krogh, A.; Mian, I. S.; Sjolander, K.; and Haussler, D. 1993. Using dirichlet mixture priors to derive hidden markov models for protein families. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*.

Chothia, C. 1992. One thousand families for the molecular biologist. *Nature* 357:543–544.

Doolittle, R. F.; Feng, D. F.; Johnson, M. S.; and McClure, M. A. 1986. Relationships of human protein sequences to those of other organisms. *Cold Spring Harbor Symp. Quant. Biol.* 51:447–455.

Hoboem, U.; Scharf, M.; Schneider, R.; and Sander, C. 1992. Selection of a representative set of structures from the Brookhaven protein data bank. *Protein Science* 1:409–417.

Jelinik, F.; Lafferty; and Mercer, R. 1990. Basic methods of probabilistic context free grammars. *IBM Research Reports* RC16374(#72684).

Joshi, A. K.; Levy, L.; and Takahashi, M. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10:136–163.

Levinson, S. E.; Rabiner, L. R.; and Sondhi, M. M. 1983. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *The Bell System Technical Journal* 62(4).

Mamitsuka, H., and Yamanishi, K. 1992. Protein secondary structure prediction based on stochastic-rule learning. In *The Proceedings of the Third Workshop on Algorithmic Learning Theory*.

Rabiner, L. R., and Juang, B. H. 1986. An introduction to hidden markov models. *IEEE ASSP Magazine* 3(1):4–16.

Rost, B., and Sander, C. 1993. Prediction of protein secondary structure at better than 70 % accuracy. *J. Mol. Biol.* 232:584–599.

Sakakibara, Y.; Brown, M.; Underwood, R. C.; Mian, I. S.; and Haussler, D. 1994. Stochastic context-free grammars for modeling RNA. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, volume V.

Sander, C., and Schneider, R. 1991. Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.* 9:56–68.

Schabes, Y. 1992. Stochastic lexicalized tree adjoining grammars. In *Proceedings of COLING-92*.

Searls, D. B. 1993. The computational linguistics of biological sequences. In Hunter, L., ed., *Artificial Intelligence and Molecular Biology*. AAAI Press. chapter 2.

Vijay-Shanker, K., and Joshi, A. K. 1985. Some computational properties of tree adjoining grammars. In *23rd Meeting of A.C.L.*