16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

MINIMUM DESCRIPTION LENGTH BASED PROTEIN SECONDARY STRUCTURE PREDICTION

Andrea Hategan and Ioan Tabus

Institute of Signal Processing, Tampere University of Technology
P.O. Box 553, FIN-33101 Tampere, Finland
phone: +(358)331153974, fax: +(358)331153817, email: andrea.hategan@tut.fi, ioan.tabus@tut.fi
web: www.cs.tut.fi/∼hategan, www.cs.tut.fi/∼tabus

ABSTRACT

This paper introduces a new algorithm for predicting the secondary structure of a protein based on the protein's primary structure, i.e. its amino acid sequence. The problem consists in finding the segmentation of the initial amino acid sequence, where each segment carries the label of a secondary structure, e.g., helix, strand, and coil. Our algorithm is different from other existing probabilistic inference algorithms in that it uses probabilistic models suitable for directly encoding the joint information represented by the pair (amino acid sequence, secondary structure labels), and chooses as winner the secondary structure sequence providing the minimum representation, or description length, in line with the minimum description length principle. An additional benefit of our approach is that we provide not only a secondary structure prediction tool, but also a tool that is able to compress in an efficient manner the joint sequences that define the primary and secondary structure information in proteins. The preliminary results obtained for prediction and compression show a good performance, which is better in certain aspects than that of comparable algorithms.

1. INTRODUCTION

Proteins are essential molecules to sustain life in all living organisms. The process by which a protein is folding in its three dimensional shape is a prerequisite for a protein to be able to perform its biological function. Because many diseases are the consequence of some proteins failure to adopt their 3-D functional shape (i.e. protein misfolding [1]) it is important to understand the rules by which the proteins fold in the three dimensional shape.

The three dimensional shape of a protein, known as the *tertiary structure*, is induced by its *primary structure*, i.e. the amino acid sequence, and the environmental condition. It is a lot easier to experimentally determine the primary structure of a protein than it is to determine its tertiary structure, and thus the methods for predicting the tertiary structure from the primary structure are a current topic of research. Presently, there is a huge gap between the number of proteins for which the primary structure is known and the number of proteins for which the tertiary structure, and thus their function, is known.

As a precondition for folding in the three dimensional shape, proteins form some local conformations, e.g. alpha-helices and beta-strands, called the secondary structure, which finally are essential for acquiring the three dimensional shape. Because these local conformations impose geometrical constraints on the three dimensional shape of a protein, the prediction of the secondary structure from the amino acid sequence is also an important stage in the process of predicting the tertiary structure. In this paper the aim is to predict such local conformations from the amino acid sequence, i.e. for each amino acid in a given protein sequence to predict a class label that specifies what is the type of secondary structure to which it belongs.

The field of protein secondary structure prediction from amino acid sequences has developed during four decades resulting in a large number of methods which can be classified in three generations [2]. The first generation methods, that appeared in 1960s and 1970s, evaluated the likelihood of different amino acids to form different local conformations. The second generation methods were dominating until early 1990s and used the likelihood of groups of adjacent amino acids to form different secondary structures. The third generation methods, of the last decade, additionally consider the evolutionary information contained in multiple alignments for refining the prediction of the secondary structure. A multiple alignment can be built between several homologous proteins, i.e. proteins that have similar known function, but have different amino acid sequences. Using such an alignment, one can build distribution profiles for each position in the sequence, thus providing a stronger discriminative information, by accounting for the variability which does not affect the function of a protein (thus being also less important for its 3-D shape). Various predictors can be built to use the distribution profiles, e.g., based on two layer neural networks.

Although the accuracy of the methods using evolutionary information is higher than that of the methods that are using only the amino acid sequence, the prediction of the secondary structure from amino acid sequence alone is still an important task because there are many *orphan* proteins [3] for which one needs to determine the secondary structure. For an orphan protein the evolutionary information is not available, because its amino acid sequence is not significantly similar to that of proteins with known secondary structure and function.

The most investigated approach for protein secondary structure prediction is to consider a sliding window of typically 15 amino acids and to predict the class label of the central amino acid knowing only the neighboring amino acids. The main problem of this approach is that it cannot capture amino acid correlations at a greater distance than the window length. For this reason, sliding window methods have a poor prediction accuracy for beta strands, which have to form hy-

This work was supported by the Academy of Finland (application number 213462, Finnish Programme for Centers of Excellence in Research 2006-2011) and the Graduate school in Electronics, Telecommunication and Automation (GETA).

```
MWMPPRPEEVARKLRRLGFVERMAKGGHRLYTHPDGR
LLLLLLHHHHHHHHHHHLLLEEEEELLEEEEELLLLL

k=7 S={L,H,L,E,L,E,L} 𝒮={6,11,3,5,2,5,5}
```

Figure 1: Example of a protein sequence with $n = 37$ amino acids and its secondary structure annotation

drogen bonds with other beta strands situated at a larger distance than the length of the sliding window, in order to create beta sheets in the folding process.

The problem of secondary structure prediction can be formulated also as a segmentation problem, where one starts from the primary structure sequence and determines: the number of segments, the class label of each segment and the length of each segment. This approach has been already investigated in the BSPSS [4] and IPSSP [3] algorithms. In these algorithms the goal is to find the best segmentation by maximizing the posteriori probability of the segmentation given the protein sequence, via Bayesian inference.

In our algorithm, called *mdlPSSP* (minimum description length protein secondary structure prediction), the best segmentation is chosen via MDL principle [5], by selecting the hypothesis, i.e. the segmentation, that yields the shortest description length of the data and of the model parameters. The probabilistic models and the searching method used in our algorithm are different than those used in [4] and [3], being additionally suitable for real encoding of the joint primary and secondary structure information.

The paper is organized as follows: in the next section the problem of protein secondary structure prediction is formulated; the third section presents the main features, modelling, coding, and searching, of the mdlPSSP algorithm. The experimental results are presented in the fourth section, while the conclusions are drawn in the last section.

2. PROBLEM STATEMENT

We need to predict the secondary structure annotation of a given protein sequence $X^n = \{x_1 \dots x_n\}$, i.e. to find the triplet $(k, S, \mathscr{L})$ such that

$$(k, S, \mathscr{L}) = \min_{(k,S,\mathscr{L})} CL(X^n | k, S, \mathscr{L}) \qquad (1)$$

where k is the number of secondary structure segments, $S = \{s_1 \dots s_k\}$ specifies the secondary structure type for each segment, $\mathscr{L} = \{l_1 \dots l_k\}$ specifies the length of each segment such that $\sum_{i=1}^{k} l_i = n$, and $CL(X^n | k, S, \mathscr{L})$ is the cost of encoding the protein sequence X^n when the secondary structure annotation is given by $(k, S, \mathscr{L})$. An example illustrating the notations is given in Figure 1 where the amino acid sequence of a protein is shown in the first row and its secondary structure annotation in the second row. The amino acid sequence represents the input data of the algorithm. The letters in the second row specify the secondary structure class label for each amino acid in the first row. The secondary structure information can be represented by the triplet $(k, S, \mathscr{L})$ and this triplet is the output of the algorithm.

In order to be able to compute the codelength of a protein sequence X^n given its secondary structure annotation $(k, S, \mathscr{L})$ we have to choose a set of probabilistic models, that will be denoted by $\mathscr{M}$. In order to deal with sequentially computable probabilities, we are going to assume a parametric first order dependency, $p(s_i | s_{i-1}, \mathscr{M})$, for the class labels, and a parametric dependency of the amino acid probability, given their class labels, such that the criterion in (1) will decouple as:

$$CL(X^n | k, S, \mathscr{L}; \mathscr{M}) = \sum_{i=1}^{k} CL(X^{l_i} | s_i, s_{i-1}; \mathscr{M}) \qquad (2)$$

For a given (hypothesized) secondary structure segmentation, one can use a procedure to compute the cost of the secondary structure segments $CL(X^{l_i} | s_i, s_{i-1}; \mathscr{M})$, and add all results into the overall cost (2). Our algorithm will consists of a search through the space of all segmentations in order to find the one having the minimum associated cost (1). The adopted model makes the search feasible by using the dynamic programming method. The next section presents the set of models $\mathscr{M}$ used, the procedure to compute the cost of encoding a secondary structure segment $CL(X^{l_i} | s_i, s_{i-1}; \mathscr{M})$ and the searching algorithm for the best segmentation.

3. THE MDLPSSP ALGORITHM

As in any machine learning algorithm we will define the procedures for the two steps of training and testing. In the training step, the algorithm adjusts the parameters and structure within the adopted classes of parametric models, in order to acquire knowledge about the dependencies expected between secondary and primary structures. In the testing step, the algorithm is provided with some new protein primary sequences, different than the ones given in the training step, and is asked to provide the segmentation, or secondary structure, which is subsequently assessed against the true one.

For the protein secondary structure prediction problem the training step implies choosing the structure of the parametric probabilistic models $\mathscr{M}$ and adjusting the parameter values of these models based on the average codelength resulted for the observed segments of a given secondary structure class. The issues concerning the training step are discussed in subsection 3.1. In the testing step, the secondary structure of a new protein is predicted using the models developed in the training step, through a searching algorithm described in 3.2. The predicted secondary structure is then compared with the true secondary structure annotation in order to assess the performance of the algorithm.

3.1 Coding and modelling

We start by describing how the codelength of a given secondary structure segment $CL(X^{l_i} | s_i, s_{i-1}; \mathscr{M})$ is calculated. In a typical data compression framework the encoder wants to send some data to the decoder by creating a bitstream that has to contain all the information needed for the decoder to be able to recover the original data.

When the encoder has to send one secondary structure segment X^{l_i} to a decoder, the encoder uses a set of models $\mathscr{M}$ to create a bitstream that encodes the following information: the type s_i, the length l_i, and the amino acid sequence $X^{l_i} = \{x_1 \dots x_{l_i}\}$. Using the bitstream and the same set of models $\mathscr{M}$, the decoder is able to decode from the bitstream all the information needed to recover the sent secondary structure segment. The length of the resulted bitstream is the cost of encoding the given secondary structure segment using the set of models $\mathscr{M}$, i.e. $CL(X^{l_i} | s_i, s_{i-1}; \mathscr{M})$.

Let $\mathscr{M} = \{\mathscr{M}_S, \mathscr{M}_H, \mathscr{M}_E, \mathscr{M}_L\}$ be the set of following models: $\mathscr{M}_S$ specifies the transition between consecutive secondary structure labels; $\mathscr{M}_H, \mathscr{M}_E, \mathscr{M}_L$ are conditional (tree) models for the amino acid being in one of the three secondary structure types, helix H, strand E, and coil L, respectively. Then, using the arithmetic codes [6], one can encode in the bitstream all information, with the cost of segment i given by:

$$CL(X^{l_i}|s_i, s_{i-1}; \mathscr{M}) = -\log_2 P(s_i|s_{i-1}; \mathscr{M}_S) - \sum_{j=1}^{l_i+1} \log_2 P(x_j|c_j; \mathscr{M}_{s_i}) \quad (3)$$

where $\mathscr{M}_S$ is a first order Markov model that gives the probability of observing a segment of type s_i following a segment of type s_{i-1} and $\mathscr{M}_{s_i}$ is a Tree Machine (TM) [7] that gives the probability of observing the amino acid x_j in the context c_j in a segment of type s_i. The summation in (3) has upper limit $l_i + 1$ since the encoder has to notify when a segment ends, by sending the "STOP" symbol. Thus, the models $\mathscr{M}_{H,E,L}$ include along with the standard amino acids the "STOP" symbol yielding an alphabet $\mathscr{A}$ with $n_a = |\mathscr{A}| = 21$ symbols.

A TM is built for each secondary structure type. First, a maximal tree machine (MTM) is built from a training set for each type and then the maximal trees have to be pruned in order to discriminate as much as possible between the three secondary structure types. Encoding with a TM, implies selecting the context, i.e. a certain node in the tree, in which a symbol is encoded. All these issues are discussed next.

The tree machines we use here are data structures similar to those introduced by Rissanen in [7] that keep a sorted list of pairs (context, symbol). The contexts can be seen as nodes in a tree, while the symbols forming a context define a path from the root to the node. Each node in the tree stores a probability distributions of the symbols that were seen at that node.

In our application the MTM is built from a large set of short segments, where each segment has a label, H, E, or L. The construction of the MTM is explained by the following example and it is illustrated in Figure 2. Suppose that the training set is composed of two segments: *aabacb* and *abac*. First, we pad each segment with the extra symbol "$\star$", to mark their beginning and ending. Only for this example, we assume that the alphabet is $\mathscr{A} = \{a, b, c, \star\}$. Then, for each symbol in all segments we select the longest context in which it appears, i.e. the prefix of each segment up to that symbol (the left table in Figure 2). The resulted prefixes are sorted from right to left in lexical order. The contexts that will define the MTM are selected such that the prefix uniquely identifies the associated symbol or the beginning of a segment has been reached (the right table in Figure 2). The resulted contexts are arranged in a tree such that each context defines a leaf. When a context is added to the tree, the tree is climbed starting from root and following the branches defined by context symbols from right to left. The frequency of the symbol paired with the added context is increased in all nodes (contexts) encountered on the path from root to the leaf defined by the context. For example, for the pair $(\star a\underline{ba}, c)$, the resulted context is *ba* and the associated symbol is *c*. This pair is added in the MTM as follows: increase the count for *c* in the root and then take the branch *a*. At the arrived node increase the count for *c* and take the branch *b*. At the arrived leaf increase the count for *c*. Since a context defines a node

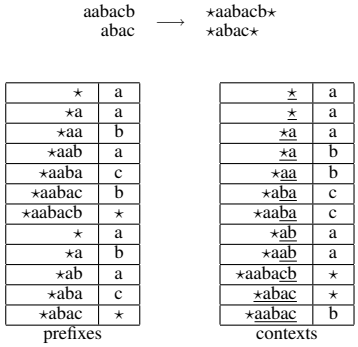

Figure 2: Maximal Tree Machine building

in the tree and a node defines a context, the two terms can be used interchangeably.

In the encoding process, the context for the current symbol x_j is selected by climbing the tree starting from the root and then taking the branches defined by the previous symbols $x_{j-1}, x_{j-2}, \ldots$ until a leaf is reached or until at the arrived node $r = x_{j-1}x_{j-2}\ldots x_{j-t}$ there is no branch with the label x_{j-t-1}. If the MTM is used to encode the training set, almost all symbols, except those at the beginning of segments, will be encoded using the distributions in leaves, where the distributions are more relevant and give better estimates for the probability of the next symbol.

If the MTM is to be used for a data set different than the training set, the so called *zero-frequency* problem will occur. This problem deals with the situation when at the arrived node r, the probability of the current symbol x_j is zero. This means that in the training phase the current symbol x_j has never been observed in the context r. Because our goal is to use the MTM for data different than the training set, we have to assign probabilities for all symbols in the alphabet at each node. This can be achieved by introducing the probability for the *escape* symbol [8], P_{esc}. In the encoding process, if in the selected context the probability of the current symbol is zero, the encoder first sends the escape symbol to inform the decoder that the current symbols has not been seen in the current context. After this, the encoder specifies which of the unseen symbols in the current context, actually is sent, using an equiprobable distribution over the unseen symbols. By introducing the probability for the escape symbol, the collected distributions at each node will be adjusted to include also this probability. Thus, if the probability of a symbol x_j in the context r is $P(x_j|r)$, the adjusted probability will be $\hat{P}(x_j|r) = (1 - P_{esc})P(x_j|r)$.

Although the longer contexts are more relevant and provide better probability estimates for the probability of the next symbol, the problem is that these longer contexts also appear less often than the shorter ones. Thus, the statistics kept in longer context accumulate slower. Even if the zero-frequency problem has been solved by introducing the escape symbol, when encoding a new data set we would like to avoid long contexts that are fitted to the training set, because here it is very likely that the current symbol will be encoded via the escape event, which yields a longer codelength than the one in a smaller context, where the symbol is more likely to be observed. Then, the goal is to prune the MTM such that to get ride of those long contexts, (over-)fitted to the training set, where encoding is inefficient.

After one maximal tree MTM_{s_i} has been created for each

of the three secondary types, $s_i = \{H, E, L\}$, the goal of pruning is to carve such trees, which will discriminate well between the three classes. From each MTM_{s_i}, we have created a set of pruned tree machines $\mathrm{TM}_{s_i}^d, d = 1 : D_{s_i}$ that keep only the nodes from root until depth level d, where D_{s_i} is the maximum level reached over the training set. The problem now is to find the optimum level $d_{s_i}^*$ yielding the most discriminative collection of three trees. The resulting trees, $\mathrm{TM}_{s_i}^{d_{s_i}^*}$, and the distribution collected at their nodes represent the abstract models denoted earlier $\mathscr{M}_{s_i}$.

The calibration process was carried out on a calibration set, that is different from both the training and testing sets. Denoting by C_{s_i} the calibration set containing all amino acid segments for type s_i, the best value of $d_{s_i}^*$ was selected by the minimization :

$$d_{s_i}^* = \min_j \{CL(C_{s_i}|\mathrm{TM}_{s_i}^j) + \sum_{q \neq s_i} [CL(C_{s_i}|\mathrm{TM}_{s_i}^j) - CL(C_q|\mathrm{TM}_{s_i}^j)]\} \quad (4)$$

With all the information described, we are able now to compute the codelength of a given secondary structure segment (3) and further to compute the codelength of a given protein sequence X^n given its secondary structure annotation $(k, S, \mathscr{L})$ (2).

3.2 Searching

In the previous sections we described how to encode a given protein sequence X^n when its secondary structure annotation $(k, S, \mathscr{L})$ is also given, based on the set of proposed models $\mathscr{M}$. Next we move to the problem of actually finding the secondary structure annotation $(k, S, \mathscr{L})$ such that the resulted codelength is minimized (1). The exhaustive search through all possible segmentations of the given protein sequence for choosing the minimizer in (1) is too expensive or even infeasible for most of the protein sequences, which have commonly hundreds of amino acids. For this reason we resort to dynamic programming, which can be applied here due to the separability of the criterion in (2). We have to define the states and their associated costs, as well as the costs for a transition from one state to another.

For minimizing (2), a state can be defined as $Z(j, k, s)$, where j represents the position of the current amino acid x_j in the given protein sequence X^n, k represents the number of segments up to position j, and s represents the secondary structure label for the segment that ends at position j. Each such state has associated a cost that represents the codelength of the protein sequence X^j, given that there are k segments and the last one has the label s. From the state $Z(j, k, s)$, the algorithm can make a transition to the new state $Z(t, k+1, p)$, with the transition cost given by $CL(X^{j+1:t}|s, p; \mathscr{M}_p)$. For each state we mark the state from which we arrived with the minimum overall cost (cost of starting state plus cost of transition). For $j = n$, there are $(n - k_{min}) * 3$ states, where $k_{min} = \lceil n/L_{max} \rceil$ and L_{max} is the maximum allowed segment. When arriving at the states with $j = n$, we select the one with the minimum cost and the best segmentation is found by following the pointers kept at each state, until the state $Z(0, 0, 0)$ is reached.

4. EXPERIMENTAL RESULTS

In order to assess the performance of any prediction algorithm, the training set and the testing set should be distinct. For the protein secondary structure prediction problem, the requirements are traditionally more stringent, i.e. the two data set should not contain sequences that share significant sequence similarity. This condition is necessary since keeping the sequences that share a certain degree of sequence similarity (likely to be homologous proteins) will artificially lower the reported prediction errors.

In our experiments, we have used three data sets. For training, we have used the PSIPRED data set from *http://bioinf.cs.ucl.ac.uk/downloads/psipred/old/data/*, for the calibration process we have used the EVA data set from *ftp://cubic.bioc.columbia.edu/pub/eva/unique_list.txt* and for testing we have used the CASP6 targets data set *http://www.predictioncenter.org/casp6/targets/cgi/casp6-view.cgi?loc=predictioncenter.org;page=casp6/*. The training and testing sets are the same as in [3] for comparison purposes.

All the data sets contain the following information for each protein: the sequence identifier, the amino acid sequence and the secondary structure annotation as defined by DSSP [9]. DSSP defines eight secondary structure classes (H,I,G,E,B,S,T,-), but usually the eight classes are mapped to three classes as suggested in [10]: (H,G)⇒H, (E,B)⇒E and (I,S,T,-)⇒L, which we adopt also here.

The performance of the prediction methods were compared in terms of per state prediction accuracy defined as $Q_i(\%) = N_i^{pred}/N_i^{obs} * 100$, where $i = \{H, E, L, 3\}$ with $i = 3$ meaning the prediction over all three states, N_i^{obs} is the number of observed amino acids in the secondary structure type i and N_i^{pred} is the number of amino acids correctly predicted in state i.

The performance of the new algorithm when working as a compression algorithm, i.e. when the secondary structure is given, was evaluated in terms of needed bits per "symbol". Here, "symbol" is taken as the pair: amino acid and secondary structure label. In this case, one needs $\log_2 20 + \log_2 3 = 5.9$ bits per symbol on average to encode a protein sequence and its secondary structure information. Since the secondary structure sequence is highly correlated, we also consider two simple alternatives. In the first one each amino acid is encoded using $\log_2 20$ bits per amino acid, while for each secondary structure segment the class label is encoded using $\log_2 3$ and the length is encoded by Elias Gamma codes for integers. The second method, transforms the secondary structure annotation in a string of 0,1 and 2. The 0 is used to mark that the class label does not change, while 1 and 2 are used to specify where the class label changes and which is the new label. The resulted string is encoded using the arithmetic codes [6]. The first method is denoted "Clear1" in Table 2, while the second is denoted "Clear2".

Table 1 presents the results obtained in the calibration process on the EVA data set. The table is split in three parts, one for each secondary structure type. The second column presents the average codelength when the segments from one class have been encoded with different pruned trees resulted from the maximal trees trained on the same class (HH,EE,LL). The third and the forth column represent the average codelength when the segments from one class are encoded using trees trained on different classes. The expression in (4) tells to choose the level that yields the minimum value in the last column, for each secondary structure type.

$\mathbf{d_H}$	HH	HE	HL	HH + (HH-HE) + (HH-HL)
1	4.82	5.92	5.79	2.73
2	4.73	6.02	5.93	2.22
3	**4.65**	7.61	6.99	**-0.64**
4	4.81	7.83	7.15	-0.54
5	4.90	7.85	7.16	-0.33
6	4.89	7.85	7.16	-0.33
7	4.90	7.85	7.16	-0.32
$\mathbf{d_E}$	EE	EH	EL	EE+(EE-EH)+(EE-EL)
1	5.53	4.98	5.66	5.94
2	**5.28**	5.04	5.54	5.25
3	5.31	5.22	5.67	**5.05**
4	5.49	5.31	5.79	5.37
5	5.56	5.33	5.81	5.53
6	5.56	5.33	5.81	5.55
$\mathbf{d_L}$	LL	LH	LE	LL+(LL-LH)+(LL-LE)
1	5.41	5.02	5.84	5.37
2	**5.21**	5.05	5.67	**4.91**
3	5.29	5.08	5.86	4.93
4	5.42	5.17	6.01	5.09
5	5.50	5.20	6.04	5.25
6	5.51	5.20	6.04	5.27

Table 1: Calibration results on EVA set

Algorithm	$\mathbf{Q_H}(\%)$	$\mathbf{Q_E}(\%)$	$\mathbf{Q_L}(\%)$	$\mathbf{Q_3}(\%)$
PSIPRED	76.06	**52.03**	69.02	67.68
BSPSS	75.17	41.74	72.69	66.54
IPSSP	74.98	46.08	**73.75**	**67.89**
mdlPSSP 322	**78.75**	21.70	52.21	57.02

Table 2: Prediction accuracy results: per state accuracy.

Table 2 presents the results for the prediction accuracy. For the mdlPSSP algorithm, the best results were obtained when the pruned trees had: 3 levels for H, 2 levels for E and 2 levels for L. Note that from Table 1, this corresponds to the best results when the segments of a certain type are encoded with the tree trained on the same secondary structure type (column 2). The results in Table 2 in the first three lines, were taken from [3]. PSIPRED is a prediction algorithm that uses additionally evolutionary information, but for the results in this table it was tested under single-sequence condition. The new algorithm mdlPPSS has the best performance for the helix class. The results are preliminary since we continue to experiment with more refined pruning techniques. On average, the prediction takes 22 seconds for one sequence on the CASP6 test set.

Table 3 presents the results for the mdlPSSP algorithm, when it works as a compression algorithm for a protein sequence and its secondary structure information. The results are better for both data sets when compared with two methods presented earlier for the "clear representation". The column "mC" represents the cost for the set of models $\mathscr{M}$ if these should also be sent to the decoder. This is compulsory for the PSIPRED data set, because the models were trained on the same set. For the EVA set, it can be assumed that the models are generic, known to both the encoder and the decoder. The column "sC" shows the average codelength obtained for encoding a protein sequence and its secondary structure annotation. The column denoted by "tC" represents the total cost, for the model and the data. The last two methods show the average codelength obtained for the two "clear representations". On average, the compression takes 0.08 seconds for one sequence on the EVA set and 0.07 seconds for a sequence in the PSIPRED data set. The compression results are compared with the clear representation because, to our knowledge, there is no other available algorithm for jointly compressing individual proteins and their secondary structure information. In a previous work, we took advantages of the secondary structure annotation in a compression algorithm for amino acid sequences, [11], but in that algorithm we targeted compression of full proteomes, which are the collection of all protein sequences in a given organism. The proteomes have length in the range of hundred thousand amino acids, and the approximate repetitions have a key role for achieving compression, which cannot be exploited in the current application, so the method from [11] cannot be applied to the present dataset.

Data	#seqs	mC	sC	tC	Clear 1	Clear 2
EVA	2181	0.13	4.86	**4.99**	5.42	5.38
PSIPRED	2242	0.13	4.83	**4.96**	5.39	5.35

Table 3: Compression results: bits per symbol (amino acid and secondary structure label).

5. CONCLUSIONS

This paper introduces a new algorithm, mdlPSSP, useful in two distinct applications: protein secondary structure prediction and joint compression of primary and secondary structure of proteins. The preliminary prediction results show better performance only for one of the classes, but the current pruning technique is still not well adapted for prediction and a more refined pruning may lead to a better performance for all classes. The algorithm can work also as a compression algorithm for individual protein sequences and their secondary structure information, being the first such algorithm for these types of data. The compression performance is significantly better when compared to a simple in clear representation.

REFERENCES

[1] C.M. Dobson, "Protein folding and misfolding," *Nature*, vol. 426, pp. 884–890, Dec. 2003.

[2] B. Rost, "Review: protein secondary structure prediction continues to rise," *Journal of Structural Biology*, vol. 134, pp. 204–218, 2001.

[3] Z. Aydin, Y. Altunbasak, and M. Borodovsky, "Protein secondary structure prediction for a single sequence using hidden semi-markov models," *BMC Bioinformatics*, vol. 7, no. 178, 2006.

[4] S.C. Schmidler, J.S. Liu, D.L. Brutlag, "Bayesian segmentation of protein secondary structure," *Journal of Computational Biology*, vol. 7, no. 1/2, pp. 233–248, 2000.

[5] J. Rissanen, "Modelling by the shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[6] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Journal of Research and Development*, vol. 20, no. 3, pp. 198–203, 1976.

[7] J. Rissanen, "A universal data compression system," *IEEE Trans. on Information Theory*, vol. IT-29, no. 5, pp. 656–664, 1983.

[8] J. Rissanen, "A lossless data compression system," *US Patent*, no. 7028042, 2006.

[9] W. Kabsch, C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, pp. 2577–2637, 1983.

[10] B. Rost, C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *Journal of Molecular Biology*, vol. 232, 584–599, 1993.

[11] A. Hategan, I. Tabus, "Jointly encoding protein sequences and their secondary structre annotation," in *Proc. GENSIPS 2007*, Tuusula, Finland, June 10-12. 2007.