

## Protein Secondary Structure Prediction Using RT-RICO: A Rule-Based Approach

By: [Leong Lee](#), Jennifer L. Leopold, Cyriac Kandath and Ronald L. Frank

L. Lee, J. L. Leopold, C. Kandath, and R. L. Frank, "Protein secondary structure prediction using RT-RICO: a rule-based approach", *The Open Bioinformatics Journal*, Volume 4, ISSN 1875-0362, pp. 17-30, 2010.

Made available courtesy of Bentham Science Publishers: <http://www.benthamscience.com/>

**\*\*\*Reprinted with permission. No further reproduction is authorized without written permission from Bentham Science Publishers. This version of the document is not the version of record. Figures and/or pictures may be missing from this format of the document.\*\*\***

### **Abstract:**

Protein structure prediction has always been an important research area in biochemistry. In particular, the prediction of protein secondary structure has been a well-studied research topic. The experimental methods currently used to determine protein structure are accurate, yet costly both in terms of equipment and time. Despite the recent breakthrough of combining multiple sequence alignment information and artificial intelligence algorithms to predict protein secondary structure, the  $Q_3$  accuracy of various computational prediction methods rarely has exceeded 75%. In this paper, a newly developed rule-based data-mining approach called RT-RICO (Relaxed Threshold Rule Induction from Coverings) is presented. This method identifies dependencies between amino acids in a protein sequence and generates rules that can be used to predict secondary structure. RT-RICO achieved a  $Q_3$  score of 81.75% on the standard test dataset RS 126 and a  $Q_3$  score of 79.19% on the standard test dataset CB396, an improvement over comparable computational methods.

**Keywords:** Data mining, Protein secondary structure prediction, Parallelization.

### **Article:**

#### **1. INTRODUCTION**

Prediction of 3D structure of a protein from its amino acid sequence is a very important research goal in biochemistry and bioinformatics, and has been studied extensively since the 1960s. Protein structure prediction is valuable for drug design, enzyme design, and many other biotechnology applications. Rost [1] suggests that although protein 3D structure prediction from sequence still cannot be achieved fully, in general, research has continuously improved methods for predicting simplified aspects of structure. Particularly in the area of secondary structure prediction, accuracy has surpassed the 70% threshold for all residues of a protein. That breakthrough was achieved by combining multiple sequence alignment information and artificial intelligence algorithms.

It is not an easy task to evaluate the performance of a protein secondary structure prediction method. For example, the use of different datasets for training and testing each algorithm makes it difficult to find an objective comparison of methods [2]. Interestingly, Kabsh and Sanders [3] tested prediction methods using proteins that had not been used in the development of the algorithms and found that the reported prediction accuracy of most of those methods decreased by more than 7%. One method's prediction accuracy decreased by as much as 27%. Rost [1] stated that "there is no value in comparing methods evaluated on different datasets."

Efforts have been made to develop standard test datasets to accurately evaluate the performance of prediction methods. Rost and Sander [4] selected a list of 126 protein domains (the RS126 set) that now constitutes a comparative standard.

Cuff and Barton [2] described the development of a non-redundant test set of 396 protein domains (the CB396 set) where non-redundancy is the case; no two proteins in the set share more than 25% sequence identity over a length of more than 80 residues [4]. They used the CB396 set to test four secondary structure prediction methods: PHD [4], DSC [5], PREDATOR [6] and NNSSP [7]. They also combined the four methods by a

simple majority-wins method, the CONSENSUS method [2]. The resulting  $Q_3$  scores for the CB396 set were 71.9% (PHD), 68.4% (DSC), 68.6% (PREDATOR), 71.4% (NNSSP) and 72.9% for the CONSENSUS method.

In the same research study, Cuff and Barton [2] also tested the RS126 set in which the  $Q_3$  scores were 73.5% (PHD), 71.1% (DSC), 70.3% (PREDATOR), 72.7% (NNSSP) and 74.8% for the CONSENSUS method.

An interesting secondary structure prediction method described by Fadime, O'zlem and Metin [8] uses a two-stage approach. In the first stage, the folding type of a protein is determined. The second stage utilizes data from the Protein Data Bank (PDB) [9] and a probabilistic search algorithm to determine the locations of secondary structure elements. The resulting average accuracy of their prediction score is 74.1%. However, the test dataset was not RS126 or CB396.

In this paper, we present a new method for predicting the secondary structure elements for different folding types. The algorithm, RT-RICO (Relaxed Threshold Rule Induction from Coverings), generates rules for discovering dependencies between protein amino acid sequences and related secondary structure elements. These rules are then used to predict protein secondary structure. The RT-RICO method performed better than previously reported methods, with a  $Q_3$  accuracy of 81.75% on the RS126 set and 79.19% on the CB396 set.

The RT-RICO approach and the main RT-RICO rule generation algorithm are discussed in Sections 3 and 4. A parallelized version of this algorithm is presented in Section 5, and detailed results of this method are presented in Section 6.

## 2. RELATED WORK

Rost [1] classifies protein secondary structure prediction methods into three generations. The first generation methods depend on single residue statistics to perform prediction. The second generation methods depend on segment statistics. The third generation methods use evolutionary information to predict secondary structure. For example, PHD [4] is a third generation prediction method based on a multiple-level neural network approach. It has been the most accurate method for many years.

One of the best secondary structure predictors is PSIPRED Protein Structure Prediction Server [10], which was developed at University College London [10, 11]. PSIPRED uses a two-stage neural network to predict the protein's secondary structure based on position-specific scoring matrices. The matrices are generated by PSI-BLAST (Position-Specific Iterated BLAST) [12]. The PSIPRED's  $Q_3$  score based on a set of 187 unique folds is between 76.5% and 78.3% [10]. There are other secondary structure prediction methods that utilize neural network prediction algorithms. For example, Jnet works by applying multiple sequence alignments alongside profiles such as PSI-BLAST and HMM [13].

Random errors in the DNA sequence lead to a different translation of protein sequences. These 'errors' are the basis for evolution [1]. Due to the fact that mutations resulting in a structural change are not likely to survive, Rost states that the evolutionary pressure to conserve structure and function has led to a record of the unlikely event: structure is more conserved than sequence [1]. Many third generation methods capitalize on this event to improve prediction accuracy. In PHD [4], Rost and Sander use multiple sequence alignments rather than single sequences as input to a neural network. At the training stage, a database of protein families aligned to proteins of known structure is used. At the prediction stage, the database of sequences is scanned for all homologues of the protein to be predicted, and the family profile of amino acid frequencies at each alignment position is fed into the network [14]. PSIPRED take advantage of the same concept, but uses a slightly different approach, *via* matrices generated by PSI-BLAST [10].

These artificial neural network methods are revolutionary in the sense that they employ the homologues of proteins for training and prediction. It is considered that a neural network is like a "black box"; it is difficult to formulate an algorithm from a neural network. A trained network may succeed in solving a problem, but it is hard to understand how it works. As a result, we are inspired to utilize a different approach, a rule-based

prediction method. This approach still makes use of the fundamental principle that structure is more conserved than sequence. We establish rules between each known secondary structure element and its “neighboring” amino acid residues. These rules are used to perform predictions. Due to the different approaches, it is difficult to directly compare prediction results between this method and other methods. Neural network methods normally employ rigorous cross-validation testing techniques. The final  $Q_3$  scores comparison should be used as a general guide, not a strict percentile comparison.

Recently, there is a trend using the support vector machine (SVM) to predict protein secondary structures. Hu, Pan, Harrison and Tai [15] achieved a  $Q_3$  accuracy of 78.8% on the RS126 dataset using a SVM approach. Kim and Park [16] developed the SVMpsi method that resulted in  $Q_3$  scores of 76.1% on the RS126 dataset and 78.5% on their KP480 dataset. Nguyen and Rajapakse [17] proposed a two-stage multi-class SVM approach utilizing position-specific scoring matrices generated by PSI-BLAST. Their  $Q_3$  scores were 78.0% on the RS126 dataset and 76.3% on the CB396 dataset.

Levitt and Chothia [18] proposed to classify proteins as four basic types according to their  $\alpha$ -helix and  $\beta$ -sheet content. “All- $\alpha$ ” class proteins consist almost entirely (at least 90%) of  $\alpha$ -helices. “All- $\beta$ ” class proteins are composed mostly of  $\beta$ -sheets (at least 90%). The “ $\alpha/\beta$ ” class proteins have alternating, mainly parallel segments of  $\alpha$ -helices and  $\beta$ -sheets. The “ $\alpha+\beta$ ” class proteins have a mixture of all- $\alpha$  and all- $\beta$  regions, mostly in sequential order. The first stage of the two stage method developed by Fadime, O’zlem and Metin [8] is able to determine the class of unknown proteins with 100% accuracy. Given a protein sequence, they use a mixed-integer linear program (MILP) approach to decide if the protein sequence belongs to one of the four classes (“all- $\alpha$ ”, “all- $\beta$ ”, “ $\alpha/\beta$ ”, or “ $\alpha+\beta$ ”). In the second stage they use a probabilistic approach based on their stage one results. The amino acid sequences of the training set are distributed into overlapping sequence groups of three to seven residues. These groups are used to calculate the probability statistics for secondary structure. Specifically, the secondary structure at a particular sequence location is determined by comparing the probabilities that an amino acid residue is a particular secondary structure type based on the statistics.

Their results are impressive. They achieved a 100% accuracy for classifying proteins into one of the four protein type classes (“all- $\alpha$ ”, “all- $\beta$ ”, “ $\alpha/\beta$ ”, or “ $\alpha+\beta$ ”). This greatly simplifies part of the protein secondary structure prediction problem. That is, given a protein amino acid sequence, if it can be determined which one of the four classes this protein belongs to, then other approaches can be applied to predict the secondary structure elements within these four classes. In contrast, the RT-RICO method uses a rule-based approach as an alternative way to make the prediction.

A study by Maglia, Leopold, and Ghatti [19] implemented a data mining approach based on rule induction from coverings in order to identify non-independence in phylogenetic data. Although rule induction from coverings appeared to be a promising solution for the phylogenetic data non-independence problem, it suffered from exponential computational complexity (which was in part addressed by a parallelized implementation that was tailored for the phylogenetic data [20]) as well as the strictness required for the resulting rules (i.e., all rules had to be correct for all instances in the dataset). The restrictive requirement for the rules is addressed in Section 3, and this allowed the research team to discover meaningful rules in another problem domain, protein datasets.

Kabsch and Sander developed a set of simple and physically motivated criteria for secondary structure, programmed as a pattern-recognition process of hydrogen-bonded and geometrical features extracted from x-ray coordinates [21]. This DSSP (Define Secondary Structure of Proteins) algorithm is the standard method for assigning secondary structure to the primary structure (amino acids) of a protein. Depending on the pattern of hydrogen bonds, DSSP recognizes eight types or states of secondary structure. The 3-helix (3/10 helix), alpha helix, and 5 helix (pi helix) are symbolized as G, H and I, respectively. DSSP recognizes two types of hydrogen-bond pairs in beta sheet structures, the parallel and antiparallel bridge. Residue in isolated beta-bridge is symbolized by B, whereas E represents an extended strand, and participates in a beta ladder. The remaining types are T for hydrogen bonded turn, and S for bend. There is also blank or “-” meaning “loop” or “other.”

These eight types are usually grouped into three classes: helix (G, H, and I), strand/sheet (E and B) and loop/coil (all others).

### 3. RT-RICO APPROACH

#### 3.1. Problem Description

In general, the protein secondary structure prediction problem can be characterized in terms of the following components [22]:

- Input

Amino acid sequence,  $A = a_1, a_2, \dots, a_N$

Data for comparison,  $D = d_1, d_2, \dots, d_N$

$a_i$  is an element of a set of 20 amino acids,  $\{A, R, N, \dots, V\}$

$d_i$  is an element of a set of secondary structures,  $\{H, E, C\}$ , which represents helix  $H$ , sheet  $E$ , and coil  $C$ .

- Output

Prediction result:  $M = m_1, m_2, \dots, m_N$

$m_i$  is an element of a set of secondary structures,  $\{H, E, C\}$

- 3-Class Prediction [23]

This is a characterization of the problem as a multi-class prediction problem with 3 classes  $\{H, E, C\}$  in which one obtains a  $3 \times 3$  confusion matrix  $Z = (z_{ij})$ .  $z_{ij}$  represents the number of times the input is predicted to be in class  $j$  while belonging to class  $i$ .

$$Q_{total} = 100 \sum_i Z_{ii} / N$$

- Q3 Score

Accuracy is computed as  $Q_3 = W_{\alpha\alpha} + W_{\beta\beta} + W_{cc}$

$W_{\alpha\alpha}$  = % of helices correctly predicted

$W_{\beta\beta}$  = % of sheets correctly predicted

$W_{cc}$  = % of coils correctly predicted

In other words, a protein secondary structure data sequence  $D$  is compared to the prediction result sequence  $M$  to calculate the  $Q_3$  score.

#### 3.2. RT-RICO Step 1, Data Preparation

RT-RICO (Relaxed Threshold Rule Induction from Coverings) is the implementation of a prediction method for solving the protein secondary structure prediction problem. First, all protein names and corresponding folding types of each protein are retrieved from the SCOP database [24, 25]. All available corresponding protein sequences and secondary structure sequences are retrieved from the PDB database [9]. Five databases of protein domains (with their amino acid sequences and secondary structure sequences) of different protein domain types (“all- $\alpha$ ”, “all- $\beta$ ”, “ $\alpha/\beta$ ”, “ $\alpha+\beta$ ” and “others”) are built. Proteins from the test datasets (RS126 or CB396) are first removed from these databases, so that they will be excluded from the possible training datasets. Protein domains from different protein families are selected to form the training datasets. See Table 1 for the number of protein domains in each training dataset on the RS126 test dataset.

**Table 1. Protein Secondary Structure Prediction Using RT-RICO Rule Generation on RS126 Test Dataset**

Folding Type Classes	Training Set		
	Number of Protein Domains	Number of 5-Residue Segments	Number of Rules (at 90% threshold)
All- $\alpha$	9,208	1,354,981	572,531
All- $\beta$	14,524	2,056,353	576,509
$\alpha/\beta$	13,337	3,366,832	710,292
$\alpha+\beta$	13,502	2,049,211	593,094
Others	6,862	1,051,281	447,696
RS126 Test Set (126 Protein Domains)			
Folding Type Classes	Number of Residues		Q <sub>3</sub> (%)
All- $\alpha$	3,424		<b>87.40</b>
All- $\beta$	6,430		<b>82.22</b>
$\alpha/\beta$	8,108		<b>78.05</b>
$\alpha+\beta$	3,068		<b>84.64</b>
Others	2,381		<b>81.23</b>
Total	23,411		<b>81.75</b>

The protein secondary structure sequences from PDB are formed by elements of eight states of secondary structure, {H, G, I, E, B, T, S, -}. The eight states are converted to four states to facilitate rule generation as follows:

(G, H, I) => Helix H

(E, B) => Sheet E

(T, S) => Coil C

(-) => “-”

Note that rule generation uses a four-state decision at-tribute. The final Q<sub>3</sub> score calculation uses a three-state decision attribute:

(G, H, I) => Helix H

(E, B) => Sheet E

(Rest) => Coil C

The basis for the RT-RICO approach is to first search segments of amino acid sequences of known protein secondary structures, and then find the rules that relate amino acid residues to secondary structure elements. The generated rules are subsequently used to predict the secondary structure. Klepeis and Floudas [26] showed that the use of overlapping segments of five residues is very effective in predicting the helical segments of proteins. Thus, the overlapping 5-residue segments approach was used to prepare the training data records. As shown in Fig. (1), for each secondary structure element, five “neighboring” amino acid residues are extracted to form a segment of five amino acid residues, plus one secondary structure element. These segments are used as input to the RT-RICO rule generation algorithm (Section 3.3, with more detail in Section 4) to generate rules. The numbers of 5-residue segments generated for the five protein type classes are shown in Table 1.

Although we use 5-residue segments, there is no evidence that five is the best segment length for this algorithm. PSIPRED uses a window of 15 amino acid residues for the neural network design [10]. Most previous methods combine multiple sequence alignment information and machine learning techniques. The purpose is to find the highly-correlated patterns from the training databases. A challenging future research problem remaining for RT-RICO is how to choose the best residue segment length, hence extracting correct and concise rules.

The main inputs to the RT-RICO rule generation algorithm are in the form of 6-tuples. The first five elements of a 6-tuple are formed by amino acid residues, {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. The last element of a 6-tuple is formed by one of four secondary structure states {H, E, C, -}. The last element is considered the decision attribute. In other words, the input to RT-RICO Step 2, Rule Generation, is in the form of an  $m \times (n+1)$  matrix, where  $m$  is the number of all entities (the number of 5-residue plus one secondary structure element segments), and  $n = |S|$  (the number of attributes,  $n = 5$  in this case).

As shown in Fig. (1), for a protein amino acid sequence and corresponding secondary structure sequence of length  $k$ , only the secondary structure elements from the third position to position  $(k-2)$  are extracted as the 5-residue segments. In other words, the first and second positions at the beginning of the secondary structure sequence, as well as the last and second-to-last positions at the end of the secondary structure sequence, are not extracted as 5-residue segments. To handle these positions, extractions are done slightly differently, as shown in Fig. (2).

These 3-residue and 4-residue segments also are used as input to the RT-RICO rule generation algorithm to generate rules. As previously mentioned, the input to RT-RICO Step 2, Rule Generation, is in the form of an  $m \times (n+1)$  matrix, where  $m$  is the number of all entities, and  $n = |S|$  (the number of attributes, where  $n = 3$  for 3-residue segments, and  $n=4$  for 4-residue segments). The same rule generation algorithm applies to all these segments. The rules generated are used in step 3 to predict the secondary structure elements at the first and second positions, as well as the last and second-to-last positions of unknown secondary structure sequences, respectively.

For an amino acid sequence of length  $k$ ,  $(k-4)$  5-residue segments are extracted, whereas only two 3-residue segments (in the first and last positions), and two 4-residue segments (in the second and second-to-last positions) are extracted. As the extraction is done for a large number of protein domains (Table 1), the rule generation and prediction operations in later steps involve mostly 5-residue segments in terms of the training data size. Due to this reason, only 5-residue segment numbers are recorded in the prediction result tables, and only 5-residue segment numbers are considered in the algorithm time complexity that is discussed in later sections.

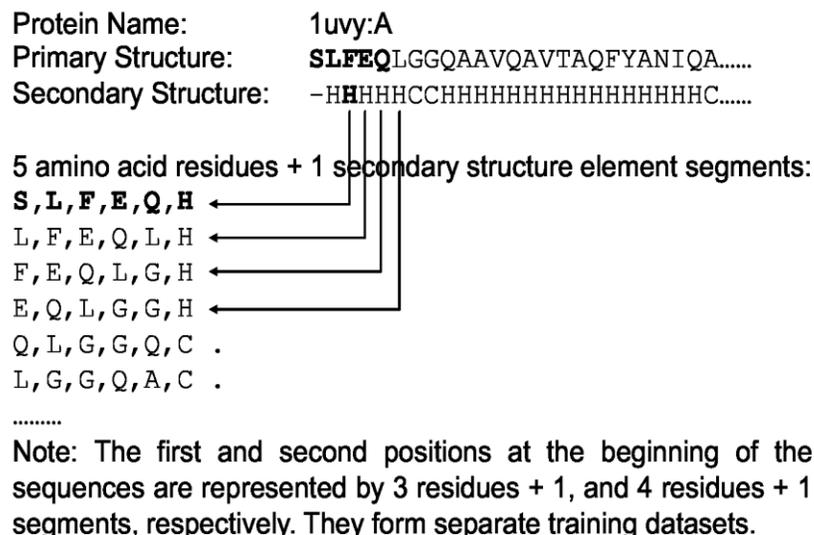
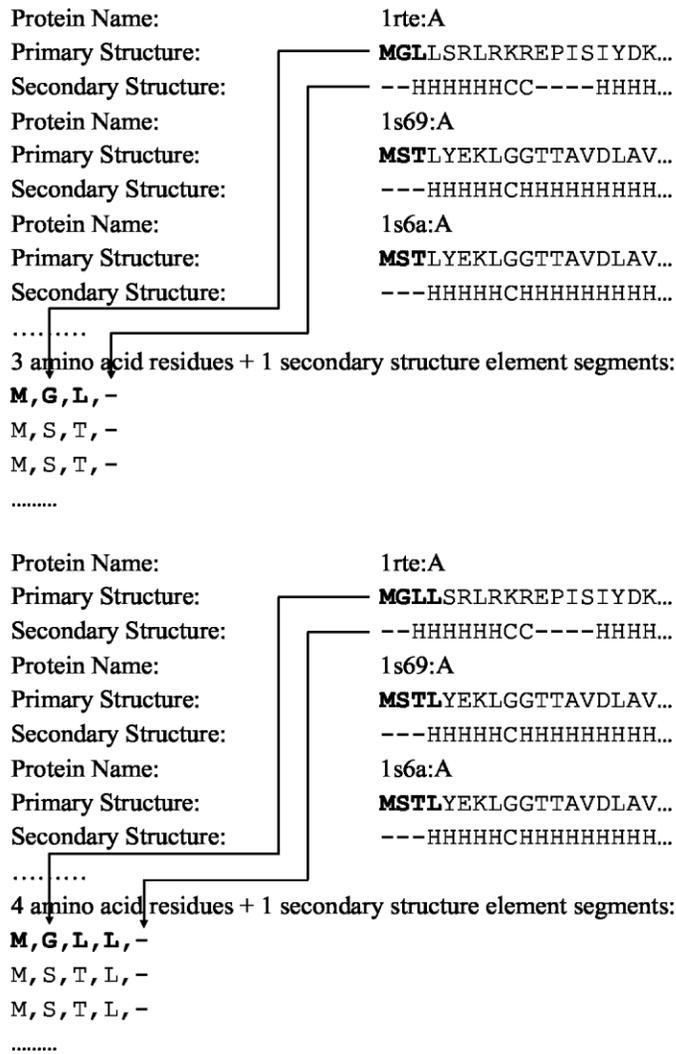


Fig. (1). Protein primary structure 5-residue segments and related secondary structure elements representation.



Note: The last and second last positions at the end of the sequences are also represented by 3 residues + 1, and 4 residues + 1 segments, respectively. The segments are generated in a similar way, but form separate training datasets.

Fig. (2). Protein primary structure 3-residue segments and related secondary structure elements representation, protein primary structure 4-residue segments and related secondary structure elements representation segments and related secondary structure elements representation, at the beginning of the sequences.

### 3.3. RT-RICO Step 2, Rule Generation

RT-RICO generates rules based on the segments in the form of an  $m \times (n + 1)$  matrix. The main RT-RICO rule generation algorithm is covered in Section 4. Some examples of the generated rules are shown in Fig. (3) in two separate formats. The first format is intended to be read by the computer programs at the later prediction stage (i.e., the computer rule format). The second format is intended to be read by the user (i.e., the human rule format). The first rule (in human rule format) is interpreted as follows: if the fourth position attribute (or “3” as interpreted by program) is “C”, and the fifth position attribute (or “4” as interpreted by program) is “C”, then the sixth attribute (decision attribute, or “5” as interpreted by program) is “H” with a confidence of 91.53% and a support of 0.04864442%. The definitions of confidence and support can be found in [27].

The corresponding first rule (in computer rule format) is interpreted as follows: if the first position attribute is “+” (representing any amino acid element), the second position attribute is “+”, the third position attribute is “+”, the fourth position attribute is “C”, and the fifth position attribute is “C”, then the sixth attribute (i.e., the decision attribute) is “H.” The number of occurrences of the fourth position attribute (which is “C”) and the fifth position attribute (which is “C”) equals 720 among all inputs to RT-RICO. The number of occurrences of the fourth position attribute (which is “C”), the fifth position attribute (which is “C”), and the sixth attribute

(which is “H”), equals 659 among all inputs to RT-RICO. The confidence is 91.53% and the support is 0.04864442%.

```

+,+,+,C,C,H,91.53,720,659,0.04864442
+,+,C,C,+,H,91.69,722,662,0.04886586
+,+,A,C,Y,H,100.00,26,26,0.00191920
.....
(3,C)(4,C) -> (5,H), 91.53%,
occurrences of ((3,C)(4,C)) = 720,
occurrences of ((3,C)(4,C) -> (5,H)) = 659, Support
% = 0.04864442
(2,C)(3,C) -> (5,H), 91.69%,
occurrences of ((2,C)(3,C)) = 722,
occurrences of ((2,C)(3,C) -> (5,H)) = 662, Support
% = 0.04886586
(2,A)(3,C)(4,Y) -> (5,H), 100.00%,
occurrences of ((2,A)(3,C)(4,Y)) = 26, occurrences
of ((2,A)(3,C)(4,Y) -> (5,H)) = 26, Support % =
0.00191920
.....

```

**Fig. (3).** Sample rules generated by RT-RICO.

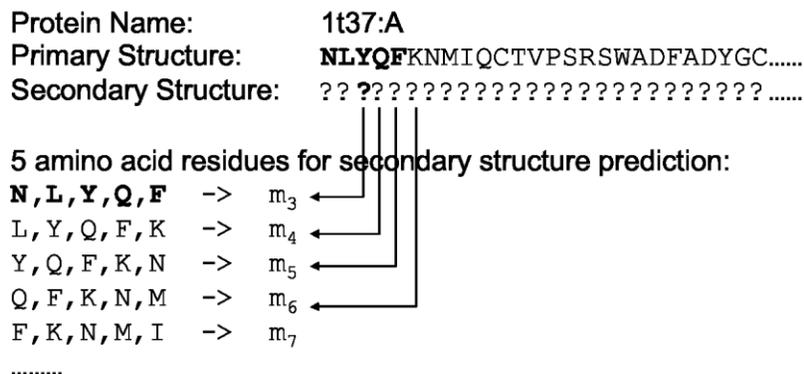
### 3.4. RT-RICO Step 3, Prediction

Finally RT-RICO loads protein primary structures from the test dataset, and predicts the secondary structure elements. As shown in Fig. (4), for each secondary structure element prediction position (for a corresponding amino acid sequence of length  $k$ , from position 3 to  $k-2$ ), five “neighboring” amino acid residues are extracted to form a segment of five amino acid residues. Each of these segments is compared with the generated rules (generated from 5-residue segments). If a segment matches a rule, the support value of the rule is taken into consideration for the prediction of the related secondary structure element.

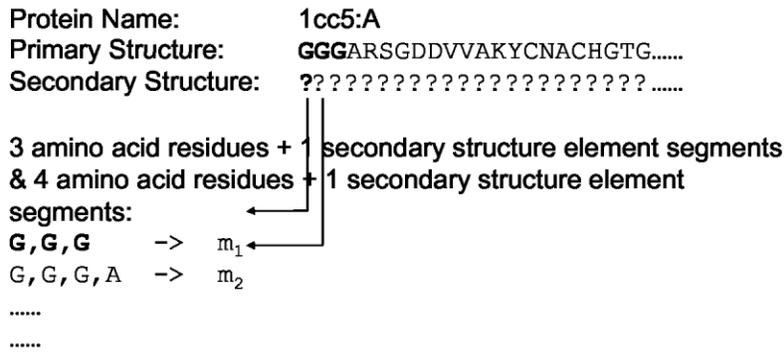
The algorithm first searches for matching rules with 100% confidence value. The secondary structure element with the highest total support value (among 100% confidence value rules) is selected.

If no matching rule exists among 100% confidence value rules, the algorithm then searches for other matching rules (with confidence values greater than or equal to 90%, but less than 100%). The secondary structure element with the highest total support value among these rules is selected as the predicted secondary structure element for that specific position.

If no matching rule is found for the segment at all, the secondary structure of the previous position is used as the predicted secondary structure.



**Fig. (4).** Protein primary structure 5-residue segments and related secondary structure elements prediction.  $m_i$  is an element of set {H, E, C, -}. It is then converted to an element of the set {H, E, C}. Note: The first and second positions at the beginning of the sequence are represented (i.e., predicted) by 3-residue, and 4-residue segments, respectively.



**Fig. (5).** Protein primary structure 3-residue, 4-residue segments, and related secondary structure elements prediction.  $m_i$  is an element of set  $\{H, E, C, -\}$ . It is then converted to an element of the set  $\{H, E, C\}$ . Note: The last and second-to-last positions at the end of the sequence are represented (i.e., predicted) by 3-residue, and 4-residue segments, respectively.

To predict the first and second positions at the beginning of a secondary structure sequence, and the last and second-to-last positions at the end of a secondary structure sequence, three or four “neighboring” amino acid residues are extracted, as shown in Fig. (5). The same prediction algorithm mentioned above is responsible for the secondary structure prediction at these positions, but instead using rules generated from 3-residue and 4-residue segments as was discussed in Section 3.2.

The number of residues used in the RS126 test dataset, and the final  $Q_3$  score of the RS126 set are shown in Table 1.

#### 4. MAIN RT-RICO RULE-GENERATION ALGORITHM

Although the RT-RICO protein secondary structure prediction method consists of the three steps mentioned in Section 3, the most computationally intensive part is in the second step, rule generation. This section covers the details of that algorithm.

##### 4.1. Rule Induction From Coverings

RT-RICO is based on a previously implemented method called RICO (Rule Induction from Coverings) [20]. RICO uses some of the concepts introduced by Pawlak [28] for rough sets, a classification scheme based on partitions of entities in a dataset [29].

In this approach, if  $S$  is a set of attributes and  $R$  is a set of decision attributes (i.e., attributes whose values we are interested in being able to determine if the values of the attributes in the set  $S$  are known), then a covering  $P$  of  $R$  in  $S$  can be found if the following three conditions are satisfied:

- i.  $P$  is a subset of  $S$ .
- ii.  $R$  depends on  $P$  (i.e.,  $P$  determines  $R$ ). That is, if a pair of entities  $x$  and  $y$  cannot be distinguished by means of attributes from  $P$ , then  $x$  and  $y$  also cannot be distinguished by means of attributes from  $R$ . If this is true, then entities  $x$  and  $y$  are said to be *indiscernible* by  $P$  (and, hence,  $R$ ), denoted  $x \sim_P y$ . An *indiscernibility relation*  $\sim_P$  is such a partition over all entities in the data set.
- iii.  $P$  is minimal.

Condition (ii) is true if and only if an equivalent condition  $\leq_2$  known as the *attribute dependency inequality*, holds for  $P^*$  and  $R^*$ , the partitions of all attributes and decisions generated by  $P$  and  $R$ , respectively, where, for a set of attributes  $A$ :

$$A^* = \prod_{a \in A} \sim [a]^*$$

The inequality  $P^* \leq R^*$  holds if and only if for each block  $B$  of  $P^*$ , there exists a block  $B'$  of  $R^*$  such that  $B$  is a subset of  $B'$ .

Once a covering is found, it is a straightforward process to induce rules from it. For example, if a set of attributes  $P = \{a_1, a_2\}$  is found to determine a set of attributes  $R = \{a_3\}$  (i.e.,  $P$  is a covering for  $R$ ), then rules of the form  $(a_1, v_1) \wedge (a_2, v_2) \rightarrow (a_3, v_3)$  (read as “if  $a_1$  equals  $v_1$  and  $a_2$  equals  $v_2$ , then  $a_3$  equals  $v_3$ ”) can be generated where  $v_1, v_2$ , and  $v_3$  are actual values of attributes  $a_1, a_2$ , and  $a_3$ , respectively, for which the relationship holds in the dataset. Such a rule also conveys a notion of non-independence between the attributes in the sets  $P$  and  $R$  (e.g.,  $a_3$  is not independent of  $a_1$  and  $a_2$ ). Here non-independence means that the relationship between the two attributes could be correlation, dependency, or co-dependency.

#### 4.2. Relaxed Attribute Dependency Inequality

All rules generated from coverings in this manner are “perfect” in the sense that there is no instance in the dataset for which the rule is not true. In order to relax this restriction somewhat (in much the same way that rules generated by decision tree induction are not always true for all instances in the dataset), the definition of the attribute dependency inequality can be modified as follows.

##### Definition 1

Relaxed Attribute Dependency Inequality

The inequality  $P^* \leq_r R^*$  holds if and only if *there exists* a block  $B$  of  $P^*$ , and *there exists* a block  $B'$  of  $R^*$  such that  $B$  is a subset of  $B'$ .

As an example for the data set of Table 2, let  $P = \{2\}$  and  $R = \{3\}$ . Then

$$\{2\}^* = \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}\}$$

$$\{3\}^* = \{\{x_1, x_2\}, \{x_3, x_5, x_6\}, \{x_4\}\}$$

There exists a block  $B = \{x_1, x_2\}$  in  $\{2\}^*$  and a block  $B' = \{x_1, x_2\}$  in  $\{3\}^*$  such that  $B \subseteq B'$ . Thus,  $\{2\}^* \leq_r \{3\}^*$  which means that  $\{3\}$  depends on  $\{2\}$  (i.e.,  $\{2\} \rightarrow_r \{3\}$ ) for at least *some* values of  $\{2\}$ . More specific rules can then be deduced from this relationship, such as  $(2, D) \rightarrow (3, H)$ .

Table 2. Decision Table with Indiscernible Relationships

	Attributes		Decision
	1 (1 <sup>st</sup> position)	2 (2 <sup>nd</sup> position)	3 (3 <sup>rd</sup> position)
$x_1$	L	D	H
$x_2$	A	D	H
$x_3$	L	C	E
$x_4$	A	C	C
$x_5$	L	R	E
$x_6$	A	R	E

#### 4.3. Relaxed Coverings

Similarly, the definition of a covering can be relaxed in order to induce rules depending on as small a number of attributes as possible.

##### Definition 2

Relaxed Coverings

A subset  $P$  of the set  $S$  is called a *relaxed covering* of  $R$  in  $S$  if and only if  $P \rightarrow_r R$  and  $P$  is minimal in  $S$ . This is equivalent to saying that a subset  $P$  of the set  $S$  is a *relaxed covering* of  $R$  in  $S$  if and only if  $P \rightarrow_r R$  and no proper subset  $P'$  of  $P$  exists such that  $P' \rightarrow_r R$ .

As an example for the dataset of Table 2, suppose rules need to be induced for  $R = \{3\}$ . The covering  $\{1, 2\}$  can be used; that is, for any assignment of values for the covering  $\{1, 2\}$ , each entity in Table 2 will induce a rule for  $\{3\}$ . But, instead of inducing a rule by looking at combinations of values for  $\{1, 2\}$ , such as  $\{1, L\} \wedge \{2, D\} \rightarrow \{3, H\}$ , rules are induced based on values for only  $\{1\}$  or  $\{2\}$ . Thus,  $\{2, D\} \rightarrow \{3, H\}$  will be generated as a rule since  $\{2\} \rightarrow_r \{3\}$  and  $\{2\}$  is minimal in  $\{1, 2\}$ . In this manner,  $\{2\}$  is a *relaxed covering* of  $\{3\}$ .

#### 4.4. Checking Attribute Dependency

To implement rule induction from coverings with the relaxed constraints, it is necessary to use the concept of checking attribute dependency, which was introduced by Grzymala-Busse [29]. In order for  $P$  to be a relaxed covering of  $R$  in  $S$ , the following conditions must be true:

- i.  $P$  must be a subset of  $S$ ,
- ii.  $R$  must depend on set  $P$  (for some values of  $P$ ), and
- iii.  $P$  must be minimal.

For the specific application of generating rules for protein secondary structure prediction, rules involving more attributes are preferred over rules involving fewer attributes, because they normally generate higher confidence values. In addition, all the possible attribute position combinations are needed to predict secondary structure. As a result, condition (iii) is not enforced for rule generation in our implementation. In fact, condition (iii) cannot be enforced for this particular application; otherwise, many meaningful rules involving multiple attributes and high confidence values would not be generated, leading to inaccurate predictions.

Condition (ii) is true if and only if the relaxed attribute dependency inequality,  $P^* \leq_r R^*$ , is satisfied.

The question then becomes how the above inequality can be efficiently checked. For each set  $P$ , a new partition, generated by  $P$ , must be determined. Partition  $U$  should be generated by  $P$ . For partitions  $\pi$  and  $\tau$  of  $U$ ,  $\pi \cdot \tau$  is a partition of  $U$  such that two entities,  $x$  and  $y$ , are in the same block of  $\pi \cdot \tau$  if and only if  $x$  and  $y$  are in the same block for both partitions  $\pi$  and  $\tau$  of  $U$ . For example, referring to Table 3,

$$\{1\}^* = (\{x_1, x_2, x_5, x_6\}, \{x_3, x_4\})$$

$$\{2\}^* = (\{x_1, x_2, x_4, x_5\}, \{x_3, x_6\})$$

$$\{1\}^* \cdot \{2\}^* = (\{x_1, x_2, x_5\}, \{x_3\}, \{x_4\}, \{x_6\})$$

**Table 3. Decision Table with Relaxed Covering**

	Attributes		Decision
	1 (1 <sup>st</sup> position)	2 (2 <sup>nd</sup> position)	3 (3 <sup>rd</sup> position)
$x_1$	L	D	H
$x_2$	L	D	H
$x_3$	A	C	E
$x_4$	A	D	C
$x_5$	L	D	H
$x_6$	L	C	-

**Table 4. Decision Table with Relaxed Covering**

	Attributes		Decision
	1 (1 <sup>st</sup> position)	2 (2 <sup>nd</sup> position)	3 (3 <sup>rd</sup> position)
$x_1$	D	A	E
$x_2$	C	A	H
$x_3$	C	A	H
$x_4$	C	A	H
$x_5$	C	A	E
$x_6$	D	A	H

That is, for  $\{1\}^*$  and  $\{2\}^*$ , two entities  $x_1$  and  $x_2$  are in the same block of  $\{1\}^*.\{2\}^*$  if and only if  $x_1$  and  $x_2$  are in the same block of  $\{1\}^*$  and in the same block of  $\{2\}^*$ . Further, the relaxed covering of  $\{3\}$  is  $\{1, 2\}$ , because  $\{1\}^*.\{2\}^* \leq_r \{3\}^*$ , and  $\{1, 2\}$  is minimal since  $\{1\}^* \leq_r \{3\}^*$  and  $\{2\}^* \leq_r \{3\}^*$  are both not true.

#### 4.5. Finding the set of All Relaxed Coverings

The algorithm R-RICO (Relaxed Rule Induction from Coverings) which is given below can be used to find the set  $C$  of all relaxed coverings of  $R$  in  $S$  (as well as the related rules).

Let  $S$  be the set of all attributes, and let  $R$  be the set of all decision attributes. Let  $k$  be a positive integer. The set of all subsets of the same cardinality  $k$  of the set  $S$  is denoted  $P_k = \{\{x_{i1}, x_{i2}, \dots, x_{ik}\} \mid x_{i1}, x_{i2}, \dots, x_{ik} \in S\}$  [29].

#### Algorithm 1: R-RICO

```

begin
  for each attribute x in S do
    compute  $[x]^*$ ;
  compute partition  $R^*$ 
k:=1
while  $k \leq |S|$  do
  for each set P in  $P_k$  do
    if  $(\prod_{x \in P} [x]^* \leq_r R^*)$  then
      begin
        find the attribute values from the
        first block B of P and from the
        first block B' of R;

        add rule to output file;
      end
    k := k+1;
end-while
end-algorithm.

```

Note that the condition (iii) for a relaxed covering is not enforced in the R-RICO algorithm. The time complexity of the R-RICO algorithm is exponential to  $|S|$ , the number of attributes in the dataset.

#### 4.6. RT-RICO Algorithm

The R-RICO algorithm produces rules that are 100% correct. However, unlike decision tree induction, R-RICO produces a more comprehensive rule set. The algorithm can be further modified to satisfy some particular level

of uncertainty in the rules (e.g., the rule is  $\geq 50\%$  true). That is, rather than just reporting a rule  $R$ , the rule can be reported as a tuple  $(R, p)$  where  $p$  is the probability that rule  $R$  is true. To accommodate this information in the rules, the definition of attribute dependency inequality must be further modified as in Definition 3.

### Definition 3

Relaxed Attribute Dependency Inequality with Threshold

Set  $R$  depends on a set  $P$  with threshold probability  $t$  ( $0 < t \leq 1$ ), and is denoted by  $P \rightarrow_{r,t} R$  if and only if  $P^* \leq_{r,t} R^*$  and there exists a block  $B$  of  $P^*$ , and there exists a block  $B'$  of  $R^*$  such that  $(|B \cap B'| / |B|) \geq t$ .

It can be observed that, when  $t = 1$ , Definitions 1 and 3 represent the same mathematical relation.

As an example, for the dataset of Table 4, let  $P = \{1, 2\}$ ,  $R = \{3\}$ , and  $t = 0.6$ . Then the following partitions can be formed:

$$\{1\}^* = \{\{x_1, x_6\}, \{x_2, x_3, x_4, x_5\}\}$$

$$\{2\}^* = \{\{x_1, x_2, x_3, x_4, x_5, x_6\}\}$$

$$P^* = \{1, 2\}^* = \{1\}^* \cdot \{2\}^* = \{\{x_1, x_6\}, \{x_2, x_3, x_4, x_5\}\}$$

$$R^* = \{3\}^* = \{\{x_1, x_5\}, \{x_2, x_3, x_4, x_6\}\}$$

There exists a block  $B = \{x_2, x_3, x_4, x_5\}$  in  $\{1, 2\}^*$ , and there exists a block  $B' = \{x_2, x_3, x_4, x_6\}$  in  $\{3\}^*$  such that  $(|B \cap B'| / |B|) = |\{x_2, x_3, x_4\}| / |\{x_2, x_3, x_4, x_6\}| = 0.75 \geq 0.6$ . Thus,  $P^* = \{1, 2\}^* \leq_{r,t} R^* = \{3\}^*$ , and  $\{3\}$  depends on  $\{1, 2\}$  (i.e.,  $\{1, 2\} \rightarrow_{r,t} \{3\}$ ), with threshold probability 0.6.

The corresponding values of attributes can be found from entities that are in  $B \cap B' = \{x_2, x_3, x_4\}$  for the sets  $P = \{1, 2\}$  and  $R = \{3\}$ ; namely, the value of attribute 1 is  $C$ , the value of attribute 2 is  $A$  at  $\{x_2, x_3, x_4\}$ , and the value of decision 3 is  $H$  for entities  $\{x_2, x_3, x_4\}$ . The rule induced from  $\{1, 2\} \rightarrow_{r,t} \{3\}$  is then  $(1, C) \wedge (2, A) \rightarrow (3, H)$  with a probability (confidence) of 75%. Another way to look at this is to note that the number of occurrences of  $((1, C)(2, A)) = 4$ , and the number of occurrences of  $((1, C)(2, A) \rightarrow (3, H)) = 3$ .

The definition of relaxed coverings must also be modified to incorporate the notion of the threshold probability given in Definition 4.

### Definition 4

Relaxed Coverings with Threshold Probability

Let  $S$  be a nonempty subset of a set of all attributes, and let  $R$  be a nonempty subset of decision attributes, where  $S$  and  $R$  are disjoint. A subset  $P$  of the set  $S$  is called a relaxed covering of  $R$  in  $S$  with threshold probability  $t$  ( $0 < t \leq 1$ ) if and only if  $P \rightarrow_{r,t} R$  and  $P$  is minimal in  $S$ .

Algorithm RT-RICO (Relaxed Threshold Rule Induction From Coverings) finds the set  $C$  of all relaxed coverings of  $R$  in  $S$  (and the related rules), with threshold probability  $t$  ( $0 < t \leq 1$ ), where  $S$  is the set of all attributes, and  $R$  is the set of all decisions. The set of all subsets of the same cardinality  $k$  of the set  $S$  is denoted  $P_k = \{\{x_{i1}, x_{i2}, \dots, x_{ik}\} \mid x_{i1}, x_{i2}, \dots, x_{ik} \in S\}$ .

## Algorithm 2: RT-RICO

```
begin
  for each attribute  $x$  in  $S$  do
    compute  $[x]^*$ ;
  compute partition  $R^*$ 
 $k:=1$ 
while  $k \leq |S|$  do
  for each set  $P$  in  $P_k$  do
    if  $(\prod_{x \in P} [x]^* \leq_{r,t} R^*)$  then
      begin
        find values of attributes from the entities
        that are in the region  $(B \cap B')$  such that
         $(|B \cap B'| / |B|) \geq t$ ;
        add rule to output file;
      end
    end
   $k := k+1$ 
end-while;
end-algorithm.
```

Note that the condition “ $P$  is minimal in  $S$ ” of a relaxed covering with threshold probability is not enforced in the RT-RICO algorithm. The reason for not implementing this condition is the same as the reason mentioned for the R-RICO algorithm. For this application, to generate rules for protein secondary structure prediction, rules involving more attributes are preferred over rules involving fewer attributes, because they normally generate higher confidence values. Also, all the possible attribute position combinations are needed for accurate prediction.

The time complexity of the RT-RICO algorithm is again exponential to  $|S|$ , the number of attributes in the dataset. Specifically, the time complexity is  $O(m^2 2^n)$ , where  $m$  is the number of all entities (the number of 5-residue segments), and  $n = |S|$  (the number of attributes). It would appear that  $2^n$  dominates the time complexity. But, for the training datasets of this application,  $n = |S| = 5$ , and  $m$  is sufficiently large. Hence,  $m^2$  dominates the time complexity in this case.

As mentioned in Section 3, the rules generated by the RT-RICO algorithm are then compared with the proteins in the test dataset to predict the secondary structure elements.

### 5. PARALLELIZED/MODIFIED RT-RICO ALGORITHM

The RT-RICO algorithm has a time complexity of  $O(m^2 2^n)$ , where  $m$  is the number of all entities (the number of 5-residue segments), and  $n = |S|$  (the number of attributes). In practice,  $n$  is only 5, while  $m$  can be fairly large. Hence,  $m^2$  dominates the time complexity. The test programs were written in PERL, and the largest  $m$  value tested was 137,715. When executed on a computer with an Intel Pentium Dual-Core processor, 2 GB of RAM, and Windows XP OS, the total program running time was approximately 14 days.

In order to accommodate a larger dataset (e.g.,  $m$  value 3,366,832), two new algorithms (Modified RT-RICO and Parallelization of Modified RT-RICO) were developed. The time complexity of modified RT-RICO is only  $O(m 2^n)$ , although it comes at an acceptable sacrifice of space complexity (i.e., more main memory space is needed, as is discussed later in this section). The program was parallelized using an NVIDIA Tesla C1060 GPU with 4GB of RAM. The 240 cores on this GPU each run at 1.3 GHz. The CPU on the same test machine is a 4-core Intel Core i7-920 with 8GB of RAM. With the modified algorithm, and the new hardware, the total program running time improved from days to a few minutes.

The focus of the parallelization of RT-RICO was the rule generation step. It is the most expensive part of the algorithm since it involves generating rules from each segment, counting the frequency of each rule, and finally calculating the confidence and support of each rule. As mentioned earlier, in the sequential implementation of RT-RICO, the complexity of this step is  $O(m^2 2^n)$ , where  $m$  is the number of segments and  $n$  is the number of amino acid residues in a segment. Usually  $n$  is fixed at 5, but  $m$  could range from a few thousand to the millions. To reduce the complexity, and hence improve its running time, it was essential to reduce the factor of  $m$  in the RT-RICO algorithm.

The  $m^2$  in  $O(m^2 2^n)$  is a result of counting the occurrences of each rule. After generating a rule from a segment, the algorithm has to iterate through the list of  $m$  segments to count how many times that rule has been seen. This has to be repeated for each of the  $m 2^n$  rules that can be generated. Hence the complexity is  $O(m^2 2^n)$ .

But RT-RICO can skip the iteration through the list  $m$  times per rule if it simply increments a rule-specific counter every time a rule is generated. The drawback is that there needs to be a counter for every possible rule that can be generated, and this requires an immense amount of main memory. In the worst-case,  $20^n \times 2^n$  rules can be generated, which translates to approximately 99 Megabytes for 5aa segments, and 163 Gigabytes for 7aa segments. This increases exponentially with an increase in  $n$ . The calculation of space complexity is illustrated in Fig. (6).

Consider a 5AA segment [ 0,1,2,3,4] and its corresponding secondary structure [ 5]

<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>4</b>

Positions 0 thru 4 can each have 20 possible amino acids, and position 5 has 4 possible secondary structures. This brings the total number of combinations to  $4 \times 20^n$ . Each of these segments can generate rules by masking the 5 amino acids in different ways. For example:

				<b>4</b>
			<b>3</b>	
			<b>3</b>	<b>4</b>
		<b>2</b>		
		<b>2</b>		<b>4</b>
		<b>2</b>	<b>3</b>	
		<b>2</b>	<b>3</b>	<b>4</b>
	<b>1</b>			
	<b>1</b>			<b>4</b>
	<b>1</b>		<b>3</b>	
	<b>1</b>		<b>3</b>	<b>4</b>
<b>...and so on</b>				

Notice how the masking of the amino acids is the same as the binary numerals for 1 thru  $2^n$ .

This means that  $2^n - 1$  rules can be generated from each segment (excluding zero).

The space required for every possible rule is:  
 $4 \times 20^n \times (2^n - 1)$  i.e.  $O(20^n \times 2^n)$

**Fig. (6).** The number of all possible rules from 5aa segments.

Despite the exponential space complexity, 5aa segments only require 99 Megabytes of memory. This was further reduced to just 4 Megabytes, by accounting for the duplicate rules that two different segments can generate. For example, the two 5aa segments [S,L,F,E,Q] and [E,L,S,E,Q] can generate the same rule for [+L+,E,Q]. The mathematics behind this space optimization is not explained here, because the 99 MB, or the 4 MB required by the modified algorithm, are both trivial amounts on the newer test machine that was used (which has 8192 Megabytes of memory).

### 5.1. Modified Algorithm for Rule Generation

In essence, the modified RT-RICO algorithm compromises on space complexity for the sake of reducing time complexity. Algorithm 3 describes this modification in more detail.

#### Algorithm 3: Modified RT-RICO

```
begin
  Allocate counters for every possible rule (initialize to 0)
  for each segment
    for each  $2^n-1$  rules that can be generated from this segment
      Calculate the memory location of the counter corresponding to this rule, and increment it
      by 1
    end-for
  end-for
  Read each counter and calculate the confidence and support for those rules that pass the relaxed threshold
end-algorithm.
```

The complexity of this algorithm is just  $O(m2^n)$  because the algorithm does not need to count the reoccurrence of each rule. The generated rules simply increment a counter whenever they are generated. There is an additional amount of time required to calculate the memory location of the counter that corresponds to a rule. However, this is negligible, and as a constant, it does not affect the overall complexity of the algorithm.

### 5.2. Parallelization of Rule Generation

The modified RT-RICO rule generation algorithm places no restrictions on the order in which rules are generated. So parallelizing the algorithm involves a straightforward distribution of the input data among processing units. Each processing unit accepts a segment as input, determines a rule from that segment, and increments the shared memory counter corresponding to that rule. Theoretically, these operations can be performed in parallel by any number of concurrent processing units. However, to minimize potentially conflicting concurrent updates of shared memory locations, the number of concurrent processing units ( $p$ ) is kept at  $2^n-1$ , which is the number of rules that a single segment can generate. Since these  $2^n-1$  rules are guaranteed to be distinct, they would guarantee mutually exclusive concurrent updates of shared memory counters. Algorithm 4 shows a parallelized version of Algorithm 3. The time complexity of Algorithm 4 is  $O((m2^n)/p)$ , where  $p$  equals the number of concurrent processing units.

#### Algorithm 4: Modified RT-RICO

```
begin
  Allocate counters for every possible rule (initialize to 0)
  for each segment s
    Send s to  $2^n-1$  processes that each calculates a different rule from it, and increment the corresponding
    shared memory counter
  end-for
  Read each counter and calculate the confidence and support for those rules that pass the relaxed threshold
end-algorithm.
```

### 5.3. Massively Parallel Computation Using GPUs

Compute Unified Device Architecture (CUDA) is a programming interface for developing general purpose applications on Graphics Processing Units (GPUs). GPUs are conventionally used for graphics acceleration, which typically involves repeatedly performing the same computational operation on multiple input data, also known as SIMD (single instruction multiple data) operations. Because of the constraints placed on SIMD operations, GPU hardware is designed with features such as massively parallel processing and pipelining to accelerate the execution of these operations. With CUDA, GPUs can be directly programmed using the C

programming language to process any kind of general purpose operation, which normally would be tasked to CPUs. However, because the GPU hardware remains the same, they are still ideally suited for SIMD operations, and more complex operations are likely to run faster sequentially on a CPU.

The modified RT-RICO rule generation algorithm is an ideal SIMD operation. The calculation of the memory location of the counter that corresponds to a rule extracted from a segment is performed over and over again for all the given segments in the input file. This SIMD operation was parallelized using an NVIDIA Tesla C1060 GPU with 4GB of RAM. The 240 cores on this GPU each run at 1.3 GHz. The CPU on the same test machine was a 4-core Intel Core i7- 920 with 8GB of RAM. The total program running time was close to 3 minutes for rule generation of the dataset in Table 1.

## 6. RESULTS

The RS126 set [4] and the CB396 set [2] are both non-redundant test datasets created with the objective of comparing different protein secondary structure prediction methods.

These two standard test datasets were used to evaluate the performance of the RT-RICO protein secondary prediction method. The two datasets have been studied extensively in other literature, and have been used as standard datasets to evaluate other prediction methods. Some of the prediction scores with different methods for the same datasets are mentioned in Sections 1 and 2. It should be noted that the CB396 set does not include protein domains from the RS126 set.

Table 1 lists the number of protein domains in each training dataset and the performance of the RT-RICO prediction method on the RS126 test dataset. Table 5 shows the number of protein domains in each training dataset and the performance of the RT-RICO on the CB396 test dataset.

**Table 5. Protein Secondary Structure Prediction Using RT-RICO Rule Generation on CB396 Test Dataset**

<b>Training Set</b>			
<b>Folding Type Classes</b>	<b>Number of Protein Domains</b>	<b>Number of 5-Residue Segments</b>	<b>Number of Rules (at 90% threshold)</b>
All- $\alpha$	9,160	1,346,571	570,580
All- $\beta$	14,466	2,046,445	574,682
$\alpha/\beta$	13,219	3,338,537	709,029
$\alpha+\beta$	13,430	2,038,220	591,909
Others	6,846	1,048,377	447,056
<b>CB396 Test Set (396 Protein Domains)</b>			
<b>Folding Type Classes</b>		<b>Number of Residues</b>	<b>Q<sub>3</sub> (%)</b>
All- $\alpha$		9,043	<b>83.50</b>
All- $\beta$		11,821	<b>80.14</b>
$\alpha/\beta$		25,909	<b>78.79</b>
$\alpha+\beta$		10,570	<b>76.50</b>
Others		3,988	<b>76.35</b>
Total		61,331	<b>79.19</b>

Cuff and Barton [2] tested the RS126 set with various prediction methods and generated Q<sub>3</sub> scores of 73.5% (PHD), 71.1% (DSC), 70.3% (PREDATOR), 72.7% (NNSSP) and 74.8% for the CONSENSUS method. The final Q<sub>3</sub> scores of RT-RICO prediction using the RS126 test dataset are shown in Table 1. The “all- $\alpha$ ” protein domains have the highest Q<sub>3</sub> score of 87.40%. The “all- $\beta$ ” and “ $\alpha/\beta$ ” protein domains have Q<sub>3</sub> scores of 82.22% and 78.05% respectively. The “ $\alpha+\beta$ ” and “Others” protein domains have the prediction accuracy of 84.64% and

81.23%. On average, RT-RICO has a  $Q_3$  score of 81.75%, which is higher than the  $Q_3$  score generated by other methods using the same RS126 test dataset reported in [2].

Cuff and Barton [2] also tested the same prediction methods using the CB396 set, resulting in  $Q_3$  scores of 71.9% (PHD), 68.4% (DSC), 68.6% (PREDATOR), 71.4% (NNSSP) and 72.9% for the CONSENSUS method. The final  $Q_3$  scores of the RT-RICO prediction method on the CB396 test dataset are shown in Table 5. The “all- $\alpha$ ” protein domains have the highest  $Q_3$  score of 83.50%. The “all- $\beta$ ” and “ $\alpha/\beta$ ” protein domains have  $Q_3$  scores of 80.14% and 78.79% respectively. The “ $\alpha+\beta$ ” and “Others” protein domains have the prediction accuracy of 76.50% and 76.35%. On average, RT-RICO has a  $Q_3$  score of 79.19%, which is higher than the  $Q_3$  score generated by other methods using the same CB396 test dataset reported in [2].

Due to the different approaches and test designs, it should be noted that it is difficult to directly compare prediction results between this method and other methods. The final  $Q_3$  scores comparison should be used as a general guide, not a strict percentile comparison.

## CONCLUSIONS

A novel rule-based method, RT-RICO, which generates rules that can be used in predicting protein secondary structure was presented in this paper. This method performed very well with the standard test datasets RS126 and CB396. The  $Q_3$  scores of 81.75% for the RS126 set and 79.19% for the CB396 set are better than the  $Q_3$  scores generated by comparable computational methods using the same datasets.

The main RT-RICO rule generation algorithm has a time complexity of  $O(m^2 2^n)$ , where  $m$  is the number of segments, with  $m^2$  dominating the time complexity. The time complexity of the modified RT-RICO algorithm is only  $O(m 2^n)$  with  $m$  dominating the time complexity, although it comes at an acceptable sacrifice of space complexity. The time complexity of the parallelized RT-RICO algorithm is  $O((m 2^n)/p)$  where  $p$  is equal to the number of concurrent processing units.

The resulting fast running time of the program enables us to generate rules from the large amount of available protein data within an acceptable timeframe, and to predict the secondary structure of available test datasets efficiently. In the future, we plan to continue to look for ways to improve the accuracy of this new promising rule-based prediction method.

## REFERENCES

- [1] B. Rost, “Rising accuracy of protein secondary structure prediction”, in *Protein structure determination, analysis, and modeling for drug discovery*, (ed. D Chasman), New York: Dekker, 2003, pp. 207–249.
- [2] J. A. Cuff, and G. Barton, “Evaluation and improvement of multiple sequence methods for protein secondary structure prediction”, *Proteins*, vol. 34, pp. 508–519, 1999.
- [3] W. Kabsh, and C. Sander, “How good are predictions of protein secondary structure?”, *FEBS Lett.*, vol. 155, pp. 179-182, 1983.
- [4] B. Rost, and C. Sander, “Prediction of protein secondary structure at better than 70% accuracy”, *J. Mol. Biol.*, vol. 232, pp. 584-599, 1993.
- [5] R. D. King, and M. J. E. Sternberg, “Identification and application of the concepts important for accurate and reliable protein secondary structure prediction”, *Protein Sci.*, vol. 5, pp. 2298–2310, 1996.
- [6] D. Frishman, and P. Argos, “Seventy-five percent accuracy in protein secondary structure prediction”, *Proteins*, vol. 27, pp. 329–335, 1997.
- [7] A. A. Salamov, and V. V. Solovyev, “Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments”, *J. Mol. Biol.*, vol. 247, pp. 11–15, 1995.
- [8] U. Y. Fadime, Y. O’zlem, and T. Metin, “Prediction of secondary structures of proteins next term using a two-stage method”, *Comput. Chem. Eng.*, vol. 32(1-2), pp. 78-88, 2008.
- [9] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, “The Protein Data Bank”, *Nucleic Acids Res.*, vol. 28, no.1, pp. 235-242, 2000.

- [10] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices", *J. Mol. Biol.*, vol. 292, no. 2, pp. 195-202, 1999.
- [11] K. Bryson, L. J. McGuffin, R. L. Marsden, J. J. Ward, J. S. Sodhi, and D. T. Jones, "Protein structure prediction servers at University College London", *Nucleic Acids Res.*, vol. 33 (Web Server issue), pp. W36-38, 2005.
- [12] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389-3402, 1997.
- [13] J. A. Cuff, and G. Barton, "Application of multiple sequence alignment profiles to improve protein secondary structure prediction", *Proteins*, vol. 40, no. 3, pp. 502-511, 2000.
- [14] B. Rost, and C. Sander, "Improved prediction of protein secondary structure by use of sequence profiles and neural networks", *Proc. Natl. Acad. Sci. USA*, vol. 90, pp. 7558-7562, 1993.
- [15] H. Hu, Y. Pan, R. Harrison, and P. Tai, "Improved protein secondary structure prediction using support vector machine and a new encoding scheme and an advanced tertiary classifier", *IEEE Trans. NanoBiosci.*, vol. 3, pp. 265-271, 2004.
- [16] H. Kim, and H. Park, "Protein secondary structure prediction based on an improved support vector machines approach", *Protein Eng.*, vol. 16, pp. 553-560, 2003.
- [17] N. Nguyen, and J. C. Rajapakse, "Two stage support vector machines for protein secondary structure prediction", *Int. J. Data Mining Bioinformat.*, vol. 1, pp. 248-269, 2007.
- [18] M. Levitt, and C. Chothia, "Structural patterns in globular proteins", *Nature*, vol. 261, no. 5561, pp. 552-558, 1976.
- [19] A. M. Maglia, J. L. Leopold, and V. R. Ghatti, "Identifying Character Non-Independence in Phylogenetic Data Using Data Mining Techniques", in *Proc Second Asia-Pacific Bioinformatics Conference Dunedin*, New Zealand, 2004.
- [20] J. L. Leopold, A. M. Maglia, M. Thakur, B. Patel, and F. Ercal, "Identifying Character Non-Independence in Phylogenetic Data Using Parallelized Rule Induction From Coverings", in *Data Mining VIII: Data, Text, and Web Mining and Their Business Applications*, WIT Transactions on Information and Communication Technologies, vol. 38, pp. 45-54, 2007.
- [21] W. Kabsch, and C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features", *Biopolymers*, vol. 22, no. 12, pp. 2577-2637, 1983.
- [22] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview", *Bioinformatics*, vol. 16, no. 5, pp. 412-424, 2000.
- [23] C. T. Zhang, and R. Zhang, "Q9, a content-balancing accuracy index to evaluate algorithms of protein secondary structure prediction", *Int. J. Biochem. Cell. Biol.*, vol. 35, no. 8, pp. 1256-1262, 2003.
- [24] A. Andreeva, D. Howorth, J. M. Chandonia, S. E. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin, "Data growth and its impact on the SCOP database: new developments", *Nucleic Acids Res.*, vol. 36 (Database issue), pp. D419-425, 2008.
- [25] G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures", *J. Mol. Biol.*, vol. 247, no. 4, pp. 536-540, 1995.
- [26] J. L. Klepeis, and C. A. Floudas, "Ab initio prediction of helical segments in polypeptides", *J. Comput. Chem.*, vol. 23, no. 2, pp. 245-266, 2002.
- [27] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2001, pp. 155-157.
- [28] Z. Pawlak, "Rough Classification", *Int. J. Man-Mach. Stud.*, vol. 20, pp. 469-483, 1984.
- [29] J. W. Grzymala-Busse, *Managing Uncertainty in Expert System*, Boston: Kluwer Academic, 1991, Ch 3.