Evolving Hidden Markov Models for Protein Secondary Structure Prediction

Kyoung-Jae Won
School of Electronics and Computer Science
University of Southampton
Southampton
SO17 1BJ, UK
j.won@ecs.soton.ac.uk

Thomas Hamelryck
University of Copenhagen
Bioinformatics Centre
Universitetsparken 15
2100 Copenhagen
thamelry@binf.ku.dk

Adam Prügel-Bennett
School of Electronics and Computer Science
University of Southampton
Southampton
SO17 1BJ, UK
apb@ecs.soton.ac.uk

Anders Krogh
University of Copenhagen
Bioinformatics Centre
Universitetsparken 15
2100 Copenhagen
krogh@binf.ku.dk

Abstract- New results are presented for the prediction of secondary structure information for protein sequences using Hidden Markov Models (HMMs) evolved using a Genetic Algorithm (GA). We achieved a Q_3 measure of 75% using one of the most stringent data set ever used for protein secondary structure prediction. Our results beat the best hand-designed HMM currently available and are comparable to the best known techniques for this problem. A hybrid GA incorporating the Baum-Welch algorithm was used. The topology of the HMM was restricted to biologically meaningful building blocks. Mutation and crossover operators were designed to explore this space of topologies.

1 Introduction

Predicting the secondary structure of proteins is one of the best studied problems in bioinformatics and consequently presents a rigorous test of our techniques. The problem tackled is to provide a label for each residue in a protein sequence depending on its secondary structure. That is, whether the protein residue is part of an alpha-helix, a beta-sheet or some other structure. This is a first step towards predicting the structure and function of a protein from its sequence. Many machine learning methods have been applied to this problem including HMMs [1], neural networks [2, 3, 4] and more recently support vector machines [5, 6].

Our approach is to this problem is to evolve an HMM using a Genetic Algorithm. An Hidden Markov Model (HMM) is a probabilistic finite state machine used to model stochastic sequences. An HMM is defined by the set of states, emission probabilities associated with each state, and transitions that connect states. One can associate a probability with a sequence according to how likely it is for an HMM to generate that sequence. To use an HMM to label a sequence we associate a label with each state (or more generally a probability of a particular label given the state). The label assigned to each element in the sequence depends on which state is likely to have emitted that element. HMMs have been widely used in bioinformatics because domain knowledge can be encoded into the topology of the HMM while still allowing other information to be learned through training the emission and transition probabilities on data. Hand-designed HMM structures have been developed for sequence alignment, protein structure prediction, gene finding and protein classifications [7]. The automation of HMM design, particularly in the context of bioinformatics, has received relatively little attention. The main reason for this is probably the desire to build domain knowledge into the topology of the HMM. This is the main attraction of HMMs over neural networks or support vector machines. However, for very complicated problems the human capacity to design meaningful HMMs becomes much more questionable and the use of automatic design becomes increasingly attractive. Over the last two years we have developed a method for evolving Hidden Markov Models (HMMs) for biological sequence analysis using Genetic Algorithms (GAs) [8, 9]. Our preliminary investigations demonstrated that we are able to obtain results that are competitive with hand-designed HMMs.

Studies of using GAs to train HMMs are relatively rare, particular in comparison with the large literature on applying GAs to neural network training. Chau *et. al* used a GA to optimise the transition and emission probabilities for a five-state HMM [10]. Subsequently, they considered evolving the HMM topology [11]. Other authors have also used GAs to find good GA topologies [12, 13]. Direct comparison of these approaches is difficult because there is not a standard test set. However, Thomsen studied the same problem of secondary structure predication as we have [13]. His prediction rate on the training set using the standards Q_3 measure is 49% compared with 75% which we achieved. We believe that the results presented here are the most impressive reported to-date on using GAs to find good HMMs for a bioinformatics problem. The results we obtain are superior to the best hand-designed HMM [1] and are competitive with the best predictor developed for this problem [2]. These results are remarkable given that the GA is competing with techniques that incorporate domain knowledge built-up over many years.

In the next section, we briefly describe the use of HMMs for labelling sequences. In section 3, we describe in detail how we evolve an HMM. In particular, we focus on unique aspects of our approach which we believe are responsible for its success. Results are presented in section 4. Finally, in section 5, we present our conclusions.

2 Hidden Markov Models with Labels

Hidden Markov Models are widely used learning machines for modelling stochastic sequences. They are described in

0-7803-9363-5/05/$20.00 ©2005 IEEE.

many papers and books on speech recognition, machine learning, and bioinformatics (see, for example, [7, 14]). In this section, we discuss how HMMs can be modified to assign a label to each element in a sequence according to its class.

2.1 Class HMM

An HMM assigns a probability to an observed sequence of symbols belonging to some alphabet, $\mathcal{S}$. We denote a sequence by $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$. In our application the symbols represent the set of 20 amino acids that are the building blocks of proteins. We denote the set of parameters that define an HMM by Θ. Given a sequence $\boldsymbol{x}$, an HMM returns a 'probability' $\mathbb{P}(\boldsymbol{x}|\Theta)$, where

$$\sum_{\boldsymbol{x}\in\mathcal{S}^T} \mathbb{P}(\boldsymbol{x}|\Theta) = 1, \tag{1}$$

so it is a probability distribution over sequences of length T (we use $\mathcal{S}^T$ to denote the set of all sequences of length T).

A class HMM (CHMM) is an HMM where the states emit class labels l from an alphabet, $\mathcal{L}$, as well as a symbol from the alphabet, $\mathcal{S}$. That is, we can associate with a sequence $\boldsymbol{x}$ a corresponding sequence of symbols $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$. Denoting the set of states by $\mathcal{Q}$, and letting $\boldsymbol{q} = (q_1, q_2, \ldots, q_T)$ be a sequence of states, then the likelihood of a sequence $\boldsymbol{x}$ with class labels $\boldsymbol{y}$ is given by

$$\mathbb{P}(\boldsymbol{x}, \boldsymbol{y}|\Theta) = \sum_{\boldsymbol{q}\in\mathcal{Q}^T} \mathbb{P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q}|\Theta) \tag{2}$$

where the sum is over all possible paths through the states (paths without transition probabilities have probability zero).

Traditionally HMMs are trained by choosing the transition and emission probabilities to maximise the likelihood of some training data. That is, given a sequence $\boldsymbol{x}$ (or set of sequence) and the corresponding labels $\boldsymbol{y}$, we find a maximum-likelihood (ML) set of parameters

$$\Theta^{ML} = \arg\max_{\Theta} \mathbb{P}(\boldsymbol{x}, \boldsymbol{y}|\Theta). \tag{3}$$

This can be calculated efficiently using the Baum-Welch algorithm [14]. However, this guarantees that we only find a local maximum-likelihood solution. As with other machine learning techniques maximising the likelihood of the training examples will not generally provide the best parameters for unseen data. This approach locally maximise the probability of the observed labelled sequences. In the case of maximising the probability of correct labelling, conditional maximum likelihood (CML) is used [15].

$$\Theta^{CML} = \arg\max_{\Theta} \mathbb{P}(\boldsymbol{y}|\boldsymbol{x}, \Theta) \tag{4}$$

where

$$\mathbb{P}(\boldsymbol{y}|\boldsymbol{x}, \Theta) = \frac{\mathbb{P}(\boldsymbol{x}, \boldsymbol{y}|\Theta)}{\mathbb{P}(\boldsymbol{x}|\Theta)} \tag{5}$$

To maximise (5) the extended Baum-Welch algorithm is used [16]. For more details on training labelled sequences see *e.g.* [17].

2.2 Posterior Label Probability

The Viterbi algorithm is a well known decoding method for HMMs. It finds the most probable path through the states of the HMM. However, in the case where there are many similar paths through the model the most probable path does not provide the best result. We rather use the 'posterior label probability' (PLP) to maximise the number of correctly predicted labels. The posterior label probability is the probability of a label at a certain position. Unlike the Viterbi algorithm it sums the probabilities of being in each state at a certain position of the sequence and assigns the dominant label to that element of the sequence.

Then the PLP of a label at position t is the sum of posterior probability of all the states that emit the same label. The PLP for label l at position t is

$$\mathbb{P}(y_t = l|\boldsymbol{x}, \Theta) = \sum_{i\in\mathcal{Q}} \mathbb{P}(y_t = l, q_t = i|\boldsymbol{x}, \Theta). \tag{6}$$

In our HMM, we assign each state to a particular class. That is, we take the probability of a label given a state to be 1 if the state is assigned to that class and 0 otherwise. Thus the sum in equation (6) only gets contributions from states that have been assigned to class l.

3 Evolving HMMs

In this section, we briefly describe the GA used to evolve HMMs for our application. We have used a hybridised GA that uses traditional GA operators to explore the space of HMM topologies in combination with Baum-Welch to optimise the transition and emission probabilities. One difficulty with evolving HMMs is that more complicated HMMs with more states and more transitions between states will always give superior results on any training set, but will typically perform badly on unseen data. One of our key aims in designing a GA for this task is to prevent this tendency to over-fit the data. We have used two strategies to accomplish this. Firstly, we impose a block structure on the HMM topology which builds in some biological plausibility. Secondly, we divide our training set into a set used for the Baum-Welch training and a set for fitness evaluation.

3.1 The Block-HMM

To constrain the search of HMM topologies to biologically meaningful structures we confine our search to a subset of topologies made up of blocks of states. We call these Block-HMMs. We used four types of blocks for the HMM: linear, self-loop, forward blocks and zero blocks. The self-loop and forward block can be either tied (we follow the convention of shading tied blocks) or untied. When a block is tied all the emission and transition probabilities inside the block are set equal. A forward block can have a transition from the first block to the n last blocks in the chain. Zero blocks have no state inside. They allows simpler structures to be explored. Figure 1 shows three types of blocks.

The block models described in this paper are motivated by applications of HMMs in biological sequence analysis.

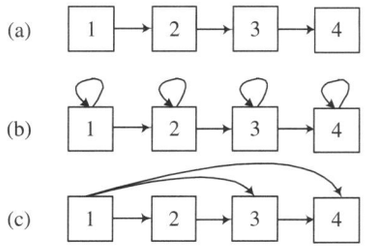

Figure 1: HMM blocks that compose the whole HMM structure. (a) linear block (b) self-loop model (tying is optional) (c) forward-jump block (tying is optional).

Biological sequences (DNA or protein) often contain "motifs", which are more or less conserved words, and with more or less homogeneous intervening sequence, which is characterised by the composition of letters (amino acids or nucleotides). Such a sequence can be modelled by an HMM containing submodels for the motifs (linear chains of states) and models for the intervening sequences if a length distribution is modelled. Other types of sequences are changing between various types of homogeneous sequences. An example is membrane proteins that contain membrane helices 20–30 amino acids long, which are dominated by hydrophobic amino acids and an intervening sequence that is typically more hydrophillic [18]. Such sequences can be modelled with a block of tied states, one block for each type of sequence. Sometimes sequences contain periodic patterns. Those HMM submodels work as a block in our scheme.

Initially, the blocks are fully linked to form the whole HMM architecture as shown figure 2. After training, most of these transition probabilities get driven to zero, so that the final structure is typically much simpler than this picture indicates.

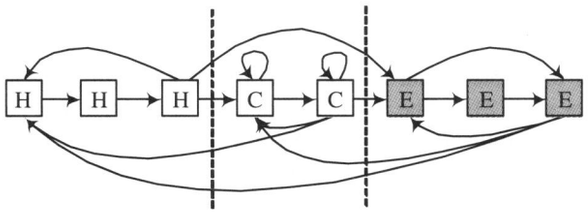

Figure 2: An example of HMM composed of blocks (Block-HMM). Three blocks are used in this model and all the blocks are fully connected to each other. For the protein secondary structure prediction each block of an HMM is assigned with one of three protein structure classes 'H', 'E', or 'C'. The definition of each label is in chapter 4.1.

To use the Block-HMM to label sequences we assign a label to each block depending on which of three protein structure classes it belongs to.

3.2 Genetic Operators for the Block-HMM

We use a GA with crossover and mutation to search the space of Block-HMM. The number of blocks is always kept fixed. However, as the blocks can have variable lengths, the number of states is not fixed. We also allow blocks consisting of no states (zero blocks), which effectively allows us to have a variable number of blocks up to some maximum.

In crossover, two parent strings are chosen at random. Some number of randomly chosen blocks are then swapped to create two children. When we swap blocks the transition probabilities leaving the block are kept as they are. Since the position of the blocks does not carry any meaning, we do not impose any constraint on which blocks are swapped. Fig. 3 shows an example of the crossover scheme. The last block of the first child crosses with the first block of the second child. To simplify the diagram, transitions between blocks are not shown here. Under the crossover scheme the properties of the interpretable blocks are not broken. This allows us to exchange meaningful blocks without causing too much disruption.

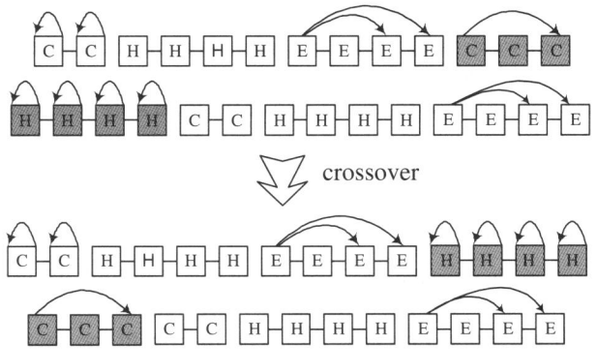

Figure 3: Crossover in Block-HMM. The crossover swaps the HMM states without breaking the property of HMM blocks.

Mutations can take place in any block of the HMM. There are a variety of different mutations that we allow. Mutations can change the length of a block. For forward-jump block mutations can change the number of transitions. For example, in the case of a 4-state forward-jump block, there are 6 different types of mutations possible. These are illustrated in Fig. 4. The outcome depends on which block is added. In the cases of linear and self-loop blocks, there is only one way to add and delete a state. These six different types of mutation supply the Block-HMM with sufficient variation without changing the properties of the block.

In addition to changing the length of the block and the transitions, we also allow another form of mutations, called *type-mutations*, that change the type of the block. For self-loop and forward jump blocks, we can mutate between tied and untied versions. We can also mutate the type altogether. Mutations to a block of zero length are also allowed. The labels are also changed under the type mutation.

3.3 Baum-Welch Training and Fitness Evaluation

At each generation we perform an iteration of Baum-Welch learning to train the emission and transition probabilities. To prevent over-fitting the learning data, we split our training data into two parts. One part is used for Baum-Welch training and the other half for the fitness evaluation. (Note,

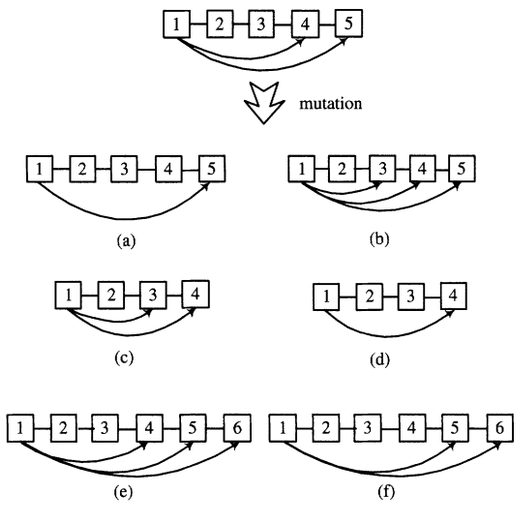

Figure 4: Six possible types of mutations from a 4-state jump forward block. (a) a transition from the first state to forth state is deleted (b) a transition from the first state to the third state is added (c) the second or the third state is deleted (d) the fourth state is deleted (e) a state is added between the fourth and the fifth state (f) a state is added between the first and the fourth state

that in evaluating the performance of our algorithm we use a completely separated dataset.) Over complex models are disadvantaged because they tend to over-fit the Baum-Welch training data, but then perform poorly on the fitness evaluation set. Splitting of the training dataset set is a unique feature of our approach which we believe is an important factor in our good performance.

We take as fitness values the reciprocal of the negative log-likelihood

$$E_\mu = \frac{1}{-\sum_i \log\left(P(x_i|\Theta_\mu)\right)/l_i} \tag{7}$$

where l_i is the length of a sequence x_i and μ labels the different HMMs (Θ_μ) of the population. Other researchers have used a penalty term in proportion to the number of states to prevent over-complex models being found. In early trials we found that this mechanism was very sensitive to the parameters being used. Although, using a penalty term may be beneficial, a penalty term which accurately punishes complexity is likely to be extremely complex function of the topology and transition and emission probabilities. Our approach of splitting the learning data allows us to side step this problem. A member of the population is selected with a Boltzmann probability

$$F_\mu = \frac{m_\mu}{\sum_{\nu=1}^{N} m_\nu}, \qquad m_\mu = \mathrm{e}^{s\,E_\mu/\sigma} \tag{8}$$

where σ is the standard deviation in the distribution of fitnesses in the population. The parameter s controls the selection strength. Stochastic universal sampling is used to reduce genetic drift in selection [19].

3.4 Parallel Genetic Algorithm

Evolving HMMs particular for this application is highly CPU-intensive. To overcome this we used a Parallel Genetic Algorithms (PGAs) [20] run on a cluster of computers. From the algorithmic point of view the parallel processing is the computational realisation of natural parallel evolutionary strategy.

We used a master-slave model to implement the Block-HMM on the clustered computers. In the master-slave model, a population is generated on P processors. The time consuming training and evaluations are dealt with on slave-processors. Because the computational time for the training and evaluation is different depending on the size of HMM, the sever waits until all the processes on the slaves finish. The slaves send the fitness value and the trained HMM to the master. Then the master applies the genetic operators on the individuals and sends the individual information back to each slave-processor. On the slave side the processor waits for message from the server, evaluates the fitness, trains the HMM, and returns the trained HMM and fitness value to the master.

4 Protein Secondary Structure Prediction

In this section, we describe the test set and the results we obtained for the prediction of the secondary structure of proteins. There is no standard test set used in this area, in part due to the constant growth of sequence data. Direct comparison between different techniques is therefore difficult. Our aim was to perform the most stringent test possible of our technique. We believe that we have made our test as difficult as possible, so that in making any comparison with other techniques the measure of performance of our technique is as conservative as possible.

4.1 Data Set

For the protein data we used the SABMark Twilight Zone data sets [21]. The Twilight Zone set is divided into sequence groups that each represent a SCOP fold. Sequence similarity is very low, between 0-25% identity, and a traceable common evolutionary origin cannot be established between most pairs even though their structures are distantly similar. This set therefore represents the worst case scenario for sequence alignment, as most related sequences share less than 25% identity. These structures belong to about 236 folds.

We removed the sequences with unresolved chain breaks and got 1662 protein sequences which are categorised to 234 folds (two folds are removed). With those 234 folds we made a five cross-validation set. As a result, sequences with the same fold do not appear in both the training and the test set.

4.2 Definition of Protein Secondary Structure

According to the DSSP [22] definition there are 8 type of structure in the protein secondary structure: H (α-helix), G (3_{10}-helix), I (π-helix), E (extended strand), B (residue in

isolated β-bridge), S (bend), T (hydrogen bonded turn) and C (others). Like most other prediction methods we used a reduction scheme whereby H and G are converted to H, E and B are converted to E, and all the other to C.

Figure 5 shows a protein sequence and its labels and predictions. The widely used Q_{index} is the percentage of residues predicted correctly as helix (Q_H), strand (Q_E), coil (Q_C) or for all three labels(Q_3). In this example Q_3 is 56% $= (14/25) \times 100$.

```
> d1ciy_2.ent
Seq     : PIRTVSQLTREIYTNPVLENFDGSF
Label   : CCCCCCCCCCEEEECHHHHCCCCCC
Predict : CCCCCHHHHHHEECCCCCCCCCCCC
```

Figure 5: A protein sequence and its label and prediction

4.3 Training with Block-HMM

We initially tested a population with 15 blocks in the HMM. We increased the number of blocks until the number of blocks does not effect the performance. The best models were found with 26–30 blocks. The initial labels are assigned evenly in order. The number of states in a block is randomly assigned to be between 1 and 4. Table 1 shows the parameters used in the simulation.

Table 1: Block-HMM parameters used in the experiment.

Parameter	value
Population size	30
Iteration	400
Number of blocks in an HMM	15–30
The length of a block	1–4
Number of crossovers per iteration	2
Number of mutations per iteration	2
Number of type-mutations per iteration	2
Ratio of evaluation data	2/7

After the training we chose the best model by evaluating the HMM with the test set. Because the GA over-fits the structure and parameters of the HMM, the best model was usually found before the end of iteration. The best model is trained again using Baum-Welch algorithm and extended Baum-Welch with the whole training dataset.

Evolving an HMM is slow. We used 31 2.4GHz P4 processors with 512Mb RAM for the simulation. Under these conditions, we could generate a new HMM within a day.

4.4 Prediction result

Figure 6 shows one of the results of Block-HMM. The simulation conducted with 25 blocks but the result showed 19 blocks because the other 6 blocks were zero blocks. Transitions between blocks are not shown here. Figure 7 is the full HMM structure with 42 states. Transitions with a probability less then 0.1 are not shown in this figure.

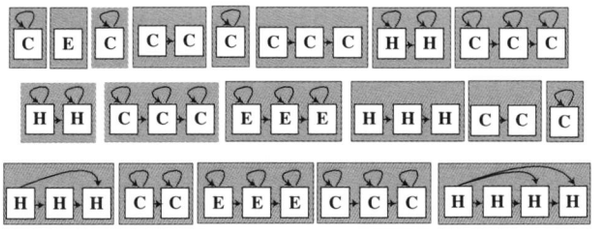

Figure 6: The result of Block-HMM. It is composed of 19 blocks and 42 states.

We used posterior label probability to decode the HMM. The most probable label is selected as an output. The result of the cross-validation test is shown in table 2. To get higher prediction rate we combined five HMMs for this simulation. Combining several HMMs improves the overall performance. But, Combining more than five HMMs did not contribute much to the performance. For the first cross-validation we reached Q_3 accuracy of 68.0% with 5 HMMs whose individual Q_3 accuracy is 67.25%, 67.34%, 67.85%, 67.72%, and 67.43%. The Q_3 accuracy of the 5-fold cross-validation test is 68.0%. It is obvious that our approach achieved far better result than Thomsen's result of 49%. From several experiments, we found that adding and deleting random transitions in the HMM does not make a significant contribution in finding efficient structural model.

Table 2: The result of 5 cross-validation test. For each test 5 HMMs are combined.

Test	Q_3			Q_H	Q_E	Q_C
	best	mean	stdev	best	best	best
Test1	68.0	67.5	0.256	64.9	56.0	75.2
Test2	70.1	68.6	0.133	65.4	59.0	75.5
Test3	67.6	66.9	0.363	67.1	52.8	74.4
Test4	68.2	67.8	0.145	67.0	58.5	74.0
Test5	67.5	66.6	0.144	64.7	55.8	74.7
Total	68.0			64.9	56.0	75.2

4.5 Multiple Sequence Alignment and Combining multiple HMMs

The best protein secondary structure predictors use homologous sequences information to gain higher prediction accuracy. We ran PSI-BLAST [23] against UniProt protein sequence database [24] to obtain homologous sequences. Each sequence from the PSI-BLAST search is weighted with position-based sequence weight [25]. In the position-based sequence weighting method the weight is given as

$$w_\kappa = \frac{\sum_{t=1}^{T} \frac{1}{r(\kappa,t)\cdot s(\kappa,t)}}{\sum_{\iota=1}^{\Gamma} \sum_{t=1}^{T} \frac{1}{r(\iota,t)\cdot s(\iota,t)}} \tag{9}$$

where Γ is the number of homologous sequence, $r(h,t)$ is the number of different residues in the position t and $s(h,t)$ is number of times the particular residue appears in the position. To exploit the homologous sequences we computed the weighted average of PLP after alignment for each

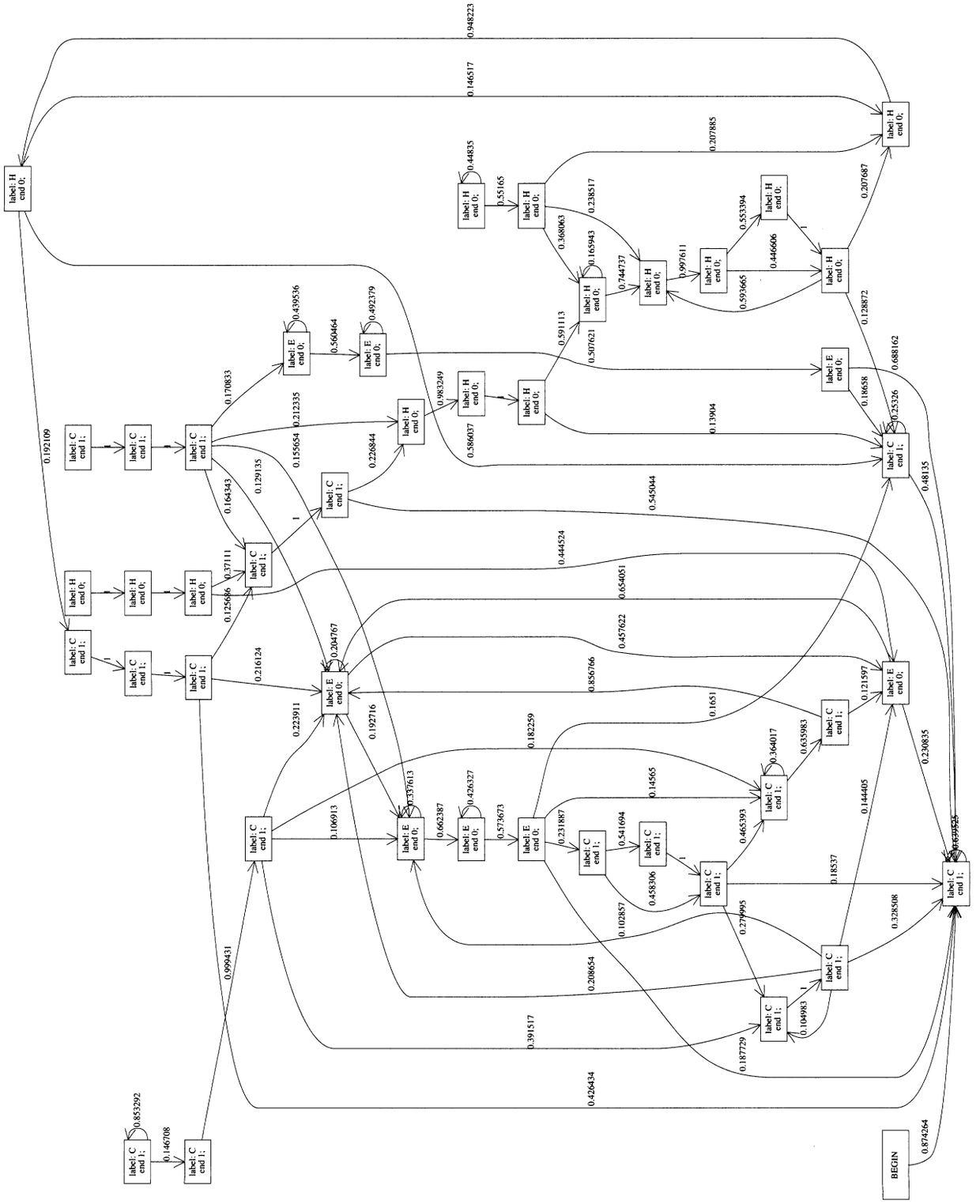

Figure 7: The full HMM structure with 42 states. Transitions less then 0.1 are not shown. This figure is drawn with the drawing tool provided by L.G.T. Joergensen.

homologue to obtained an overall average PLP. As a final enhancement we averaged the prediction over five HMMs. The final procedure is illustrated in figure 8. We used this final averaged PLP to make out prediction.

4.6 Comparison with Other Protein Secondary Structure Methods

Table 3 shows the result of 5 cross-validation tests after the multiple alignment of homologous sequences and combining 5 HMMs. The average Q_3 accuracy is 75.0% with a standard deviation of 9.12.

Table 3: The result of 5 cross-validation test after using the multiple alignment of homologous sequences and combining 5 HMMs

Test	Q_3	Q_H	Q_E	Q_C
Test1	75.3	71.7	61.0	82.4
Test2	76.6	67.0	62.6	83.0
Test3	74.8	70.5	57.1	84.5
Test4	74.7	72.3	61.1	82.3
Test5	73.8	66.4	58.1	81.9
Total	75.0	69.4	60.0	82.9

HMMSTR [1] is the most successful predictor using an HMM. It was constructed by linking I-sites (invariant or initiation sites) motifs and representing them as a chain of Markov states. Consequently, the topology of HMMSTR is a collection of known structures which are transformed into the HMM with over one hundred states. However, the prediction accuracy (Q_3) of HMMSTR is 74.3%. Even though the Block-HMM method has fewer states and a simple structure, it could produce better result.

Psipred [2] is one of the best predictor algorithms with a Q_3 prediction rate of 76.5%. It was tested under a similar stringent condition but using a 3-fold cross-validation. They used 85 test sequences and 1100 training sequences. Most of the other top predictors have similar prediction accuracy with Psipred. They are tested using sequences with less then 25% identity within the training set. Our method was tested under more stringent condition in that any fold of the sequences in the test set was not used in the training set.

5 Conclusions

In this paper, we have demonstrated the power of Genetic Algorithms to find HMM structures for biological sequence analysis. The HMM structure had considerably few states than the best hand-designed HMM and gave superior performance. This is quite remarkable given that our GA had no prior knowledge of protein structure. One of the major advantages of our approach is its simplicity. All other methods such as neural networks [2, 3, 4] or support vector machines (SVMs) [5, 6] require a large amount of tuning to reach the levels of performance we have achieved.

Relatively little work has been invested in tuning the GA. We believe there is still potential for improvement both in terms of improving the overall performance and in speeding up search by modifying our proposed algorithm. In particular, introducing other mechanisms to control the complexity of the models produce and using other block structures are likely avenue of improvement. In this work, we used reciprocal of the negative log-likelihood as a fitness function. Alternatively, the Q_3 for the evaluation set can be used directly as a fitness function. This is another future area of research.

Here is an important application where GAs can make a real contribution. Their flexibility allows us to incorporate other optimisation schemes within them and to construct operators which directs the search towards fruitful areas of the solution space. Because of the computational complexity of training HMMs this is an application area that is only now becoming practical, however, as the power of computers increase we can expect that the automatic discovery of HMMs to become increasingly attractive.

Acknowledgement

We would like to thank L.G.T. Joergensen for providing his nice HMM structure drawing tool.

References

[1] C. Bystroff, V. Thorsson, and D. Baker, "HMMSTR: a Hidden Markov Model for Local Sequence-Structure Correlations in Proteins," *Journal of Molecular Biology*, vol. 301, pp. 173–190, 2000.

[2] D. T. Jones, "Protein Secondary Structure Prediction Based on Position-specific Scoring Matricws," *Journal of Molecular Biology*, vol. 292, pp. 195–202, 1999.

[3] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, "Improving the Prediction of Protein Secondary Structure in Three and Egight Classes Using Recurrent Neural Networks and Profiles," *PROTEINS: Structure, Fuction, and Genetics*, vol. 47, pp. 228–235, 2002.

[4] K. Lin, V. A. Simossis, W. R. Taylor, and J. Heringa, "A simple and fast secondary structure prodiction method using hidden neural networks," *Bioinformatics*, vol. 21, no. 2, pp. 152–159, 2005.

[5] J. J. Ward, L. J. McGuffin, B. F. Buxton, and D. T. Jones, "Secondary structure prediction with support vector machines," *Bioinformatics*, vol. 19, no. 13, pp. 1650–1655, 2003.

[6] J. Guo, H. Chen, Z. Sun, and Y. Lin, "A Novel Method for Protein Secondary Structure Prediction Using Dual-Layer SVM and Profiles," *PROTEINS: Structure, Function, and Bioinformatics*, vol. 54, pp. 738–743, 2004.

[7] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis. Cambridge.* Cambridge University Press, 1998.

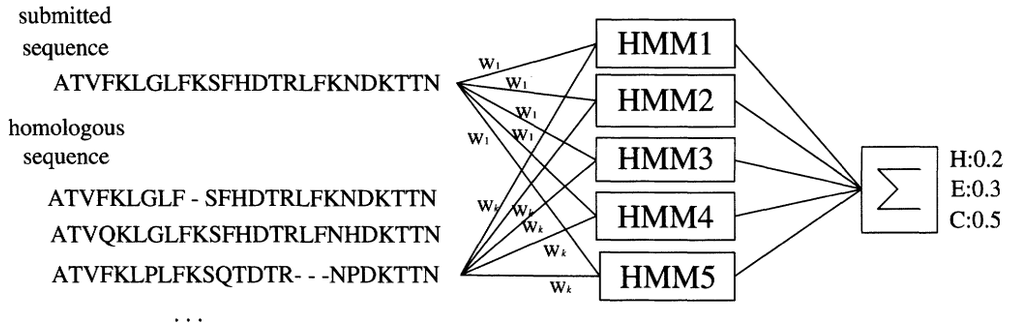

Figure 8: The protein secondary structure predictor with HMMs

[8] K.-J. Won, A. Prügel-Bennett, and A. Krogh, "Training HMM Structure with Genetic Algorithms for Biological Sequence Analysis," *Bioinformatics*, vol. 20, no. 18, pp. 3613–3627, 2004.

[9] ——, "Evolving the Structure of Hidden Markov Models," *IEEE Transactions on Evolutionary Computation*, 2005, accepted.

[10] C. Chau, S. Kwong, C. Diu, and W. Fahrner, "Optimization of HMM by a Genetic Algorithm," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. 1727–1730.

[11] S. Kwong, C. Chau, K. Man, and K. Tang, "Optimisation of HMM topology and its model parameters by genttic algorithms," *Pattern recognition*, vol. 34, pp. 509–522, 2001.

[12] T. Yada, "Stochastic models representing dna sequence data construction algorithms and their applications to prediction of gene structure and function," Ph.D. dissertation, University of Tokyo, 1998.

[13] R. Thomsen, "Evolving the Topology of Hidden Markov Models Using Evolutionary Algorithms," *LNCS*, vol. 2439, pp. 861–870, 2002.

[14] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceeding of IEEE*, vol. 77, no. 2, 1989, pp. 257–286.

[15] B. Juang and L. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991.

[16] Y. Normandin and S. Morgera, "An improved MMIE training algorithm for speaker independent, small vocabulary, continuous speech recognition," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1991, pp. 537–540.

[17] A. Krogh, "Two methods for improving performance of an HMM and their application for gene finding," in *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, 1997, pp. 179–186.

[18] A. Krogh, B. Larsson, G. von Heijne, and E. Sonnhammer, "Predicting transmembrane protein topology with a hidden Markov model: Application to complete genomes," *Journal of Molecular Biology*, vol. 305, no. 3, pp. 567–580, 2003.

[19] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates (Hillsdale), 1987, pp. 14–21.

[20] J. Cohoon, S. Hedge, W. Martin, and D. Richard, "Punctuated equilibria: A parallel genetic algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates (Hillsdale), 1987, pp. 148–154.

[21] I. V. Walle, I. Lasters, and L. Wyns, "SABmark - a benchmark for sequence alignment that covers the entire known fold space," *Bioinformatics*, vol. 21, no. 7, pp. 1267–1268, 2005.

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

[23] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucl. Acids Res.*, vol. 24, pp. 3389–3402, 1997.

[24] R. Leinonen, F. Diez, W. Fleischmann, R. Lopez, and R. Apweiler, "Uniprot archive," *Bioinformatics*, vol. 20, no. 17, pp. 3236–3237, 2004.

[25] S. Henikoff and J. Henikoff, "Position-based Sequence Weights," *Journal of Molecular Biology*, vol. 243, pp. 574–578, 1994.