A finite state automata based technique for protein classification rules induction

Fotis E. Psomopoulos
Aristotle University of Thessaloniki
48 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Pericles A. Mitkas
Aristotle University of Thessaloniki
272 PUBLICATIONS **1,635** CITATIONS
SEE PROFILE

A Finite State Automata Based Technique for Protein Classification Rules Induction

F. Psomopoulos
Dept. of Electrical and Computer Engineering, Aristotle University
54124, Thessaloniki, Greece
+30 2310 99 6349
fpsom@auth.gr

S. Diplaris
Dept. of Electrical and Computer Engineering, Aristotle University
54124, Thessaloniki, Greece
+30 2310 99 6349
diplaris@danae.ee.auth.gr

P. A. Mitkas
Dept. of Electrical and Computer Engineering, Aristotle University
54124, Thessaloniki, Greece
+30 2310 99 6390
mitkas@eng.auth.gr

ABSTRACT
An important challenge in modern functional proteomics is the prediction of the functional behavior of proteins. Motifs in protein chains can make such a prediction possible. The correlation between protein properties and their motifs is not always obvious, since more than one motifs can exist within a protein chain. Thus, the behavior of a protein is a function of many motifs, where some overpower others. In this paper a data-mining approach for motif-based classification of proteins is presented. A new classification rules inducing algorithm that exploits finite state automata is introduced. First, data are modeled by terms of prefix tree acceptors, which are later merged into finite state automata. Finally, we propose a new algorithm for the induction of protein classification rules from finite state automata. The data-mining model is trained and tested using various protein and protein class subsets, as well as the whole dataset of known proteins and protein classes. Results indicate the efficiency of our technique compared to other known data-mining algorithms.

Categories and Subject Descriptors
H.2.8 [**Information Systems**]: Database Applications – *data mining.*

General Terms
Algorithms, Performance.

Keywords
finite state automata, data mining, bioinformatics, proteomics, motifs, protein classification.

1. INTRODUCTION
Proteins are grouped into several families according to the functions they perform. All proteins contained in a family feature a certain structural relation, thus having similar properties. The biological action of proteins is traditionally identified by time consuming and expensive in vitro experiments. In recent years, bioinformatics has managed to provide a computational solution to this problem. Patterns are short amino acid chains that have a specific order, while profiles are computational representations of multiple sequence alignments using hidden Markov models. We will refer to both profiles and patterns as motifs. Motifs are widely used for the prediction of a protein's properties, since the latter are mainly defined by their motifs. The need to develop algorithms that induce rules describing these associations is obvious. A step towards this direction has already been taken, by recording such motifs in several databases, including Prosite[1], Pfam[2] and Prints[3].

The basic problem that our technique tries to solve can be stated as follows: Given a set of proteins with known properties (that have been experimentally specified), we aim to induce classification rules that associate motifs to protein families, referred to as protein classes. In this way, it is possible to classify newly discovered protein chains and define their properties. This problem can be addressed as a classic data mining problem. Using a training dataset, it is possible to create a classification rules induction model. The classification attribute is the protein class. The attributes used for data mining are the various existing motifs in each protein chain. Machine learning algorithms [4] can offer the most cost effective approach to automated discovery of a priori unknown predictive relationships from large data sets in computational biology [5]. A plethora of algorithms to address this problem have been proposed, by both the artificial intelligence and the pattern recognition communities. Some of the algorithms create decision trees [6, 7], others exploit artificial neural networks [8] or statistical models [9]. Our approach to the problem exploits finite state automata (FSA). FSAs are processors with a finite memory that can sequentially read symbols. The output they produce is an indication of whether the input string is acceptable. A requisite condition for associating motifs to protein classes is the proper representation of the protein chain. Most algorithms treat each chain element as a sorted part of a specific set of features. In our case, protein chains are represented by using a proper lexicon of motif sequences. Specifically, the sequence of motifs belonging to a protein chain describes the protein itself.

Initially, data are preprocessed and stored in a database. Then, data is transformed into a prefix tree acceptor (PTA), which is an automaton that only accepts the strings in the sample and in which common prefixes are merged together resulting in a tree-shaped automaton. Transitions represent the presence of motifs, while end states designate the presence of a protein. PTAs are constructed for every single protein class. The next step involves graph merging in all PTAs, in order to construct more compact and faster to process FSAs. A new algorithm is consequently used to extract interesting classification rules from FSAs, which are finally applied to the test dataset by order of interest.

The rest of the paper is structured as follows. We outline the steps of our methodology for rule induction and then we proceed with a

In *Proceedings of the Second European Conference on Data Mining and Text Mining in Bioinformatics*, held in conjunction with ECML/PKDD in Pisa, Italy. 24. September, 2004.

more detailed description of each step. Then the construction of the PTA is explained, while in the next chapter we present the graph merging procedure. Next, our classification rules induction algorithm is illustrated followed by statistical results from experiments that show the advantages of the algorithm. Finally, we present our conclusions and possible extensions of our work.

2. METHODOLOGY OUTLINE

The developed methodology for rule induction can be summarized as follows:

First, the train set is constructed from a set of known proteins. For every protein in the set, both the class in which it belongs and the motifs it contains must be known.

The next step is the creation of the prefix tree acceptor (PTA) using the protein chains of the train set. Using the ALERGIA algorithm [10], the PTA is converted to an equivalent stochastic finite state automaton, in which every transition is associated with a probability.

Since the FSA is a generalized representation of the protein structure, information can be obtained directly from it. As a result certain probabilities can be calculated, such as the probability of a protein chain to contain a certain motif or a specific subset of motifs. Using these probabilities, rules can be extracted to accurately describe the form of the protein chains. The rules can vary in complexity, from simple ones, such as "If motif a exists, then the protein belongs to class P", to more complex ones, such as "If motifs a and b exist, then the protein belongs either in class P1, or in class P2 or in both classes".

3. PTA CONSTRUCTION

Given a random symbol string (for example $w = a_1a_2a_3\ldots a_n$), then a prefix of the string is any subset w_p of those symbols, for which:

$$w = w_p w'$$

where w' is a string of any length, i.e. w' may contain any number of symbols, including zero.

It is obvious that, for a given set of strings, it is possible to construct a corresponding set of prefixes. In the new set, every element (prefix) will be a string with length less or at most equal to the one it came from. At the same time, however, this new set will allow the identification of a specific string using much less symbols.

These prefixes can be associated with a finite state automaton, which can identify only these strings. This automaton is called prefix tree acceptor, or PTA. Since every string is the prefix of itself, a means of creating the PTA is by using the original strings, inserting a transition between states for every symbol. Figure 1 shows an example PTA tree that represents the following strings:

$$w_1 = aaaaab$$
$$w_2 = abbaaa$$
$$w_3 = bababa$$

Obviously, the PTA can be minimized if the corresponding prefix set is used in the same way. Based on the strings above, the minimum length set of prefixes that keeps the individuality of the strings is the following:

$$w_{p_1} = aa$$
$$w_{p_2} = ab$$
$$w_{p_3} = b$$

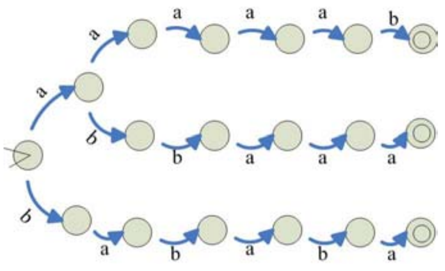

Figure 1. An example PTA tree.

Figure 2 illustrates the minimized PTA induced from the minimum length prefixes.

In the case of a protein set, the procedure is identical. The strings are no longer strings of symbols, but of a sequence of motifs. However, each motif is represented with a specific code, in the form of PSxxxxx. In this way, codes can be easily represented as symbols and the original PTA creation algorithm can be used.

The algorithm for creating a PTA from a set of strings can be stated as follows:

```
begin algorithm
  do (for every protein in the trainset)
    motifs = findProteinChain(protein)
    currentState = firstNode(PTA)
    do (for every motif m in the motif sequence)
      if (currentState does not contain transition
                                          for motif m)
        add new state newState to PTA
        add transition between currentState and
                              newState with motif m.
        currentState = newState
      else
        find state endState where currentState
                              with motif m transits.
        currentState = endState
      end if
    end do
  end do
end algorithm
```

Assuming that the train set consists of P different proteins, that every protein consists of M at most motifs, and that there are A different motifs altogether, the complexity of the algorithm is:

$$O = P\,M\,(A - 1)$$

However both A and M can be considered approximately constant. In that case, the complexity is linear to the number of elements in the train set.

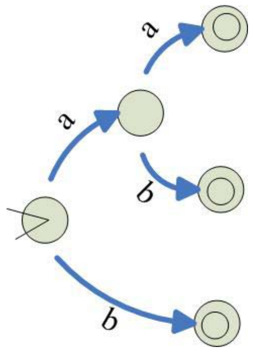

Figure 2. The minimized PTA tree.

4. MERGING PTA TO FSA

After creating the PTA, the next step is the construction of the equivalent FSA. The procedure is in fact a transformation of the PTA by means of a systematical merging of states. The algorithm that does this transformation is the ALERGIA algorithm [10].

In order for the ALERGIA to be applied, the PTA must contain certain statistical information, i.e. it must already be a stochastic FSA. This information is listed below:

1. For every state i, the number (n_i) of chains that have arrived at i must be known.
2. For every state i, the number ($f_i(\#)$) of chains that have ended in i must be known.
3. For every transition a from any state i, the number ($f_i(a)$) of chains that have used it must be known.

It is obvious that this information can be obtained during the construction of the PTA. The ratios ($f_i(a)/n_i$) and ($f_i(\#)/n_i$) show the probability $p_i(a)$ of the transition a, and the probability p_{if} of a chain ending respectively.

The ALERGIA algorithm compares states (nodes) in pairs (q_i, q_j) with $1 \le i \le j-1$ and $2 \le j \le |PTA|$, where |PTA| is the number of states in the FSA. Two states are defined to be equivalent when they have the same transition probabilities for every symbol a, and the target states are also equivalent:

$$q_i \equiv q_j \Rightarrow \begin{cases} p_i(a) = p_j(a) \\ \delta_a(i) \equiv \delta_a(j) \end{cases}, \forall a$$

However, if only state equivalence is considered, the resulting FSA will be an exact representation of the original PTA and not a generalization. For this reason, a new parameter (confidence) is used, which transforms the strict equivalence to equivalence between limits. In this case, the states are referred as compatible. The limits are defined by the Hoeffding limit for a Bernoulli variable with probability p and frequency of appearance f out of n:

$$\left| p - \frac{f}{n} \right| < \sqrt{\frac{1}{2 \cdot n} \cdot \log \frac{2}{a}}$$

with probability greater than $(1 - a)$.

Thus, a PTA describing a set of proteins (Figure 3) can be merged into a generalized, yet minimized, FSA (Figure 4), by progressively merging states (Figure 5).

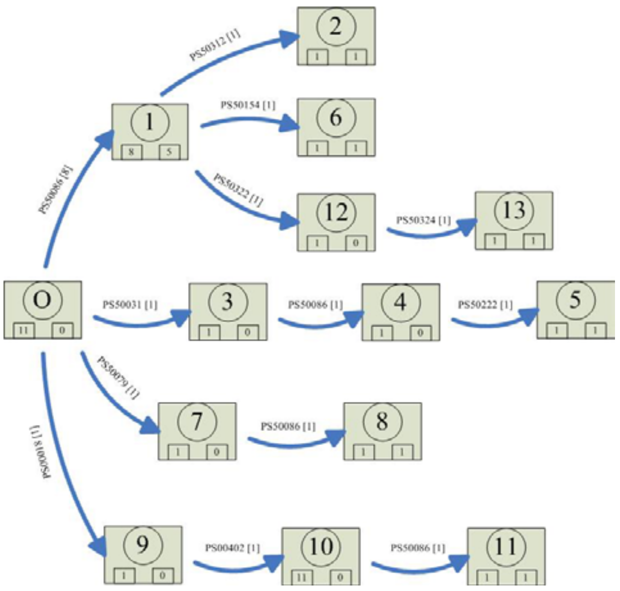

Figure 3. PTA representing a subset of proteins.

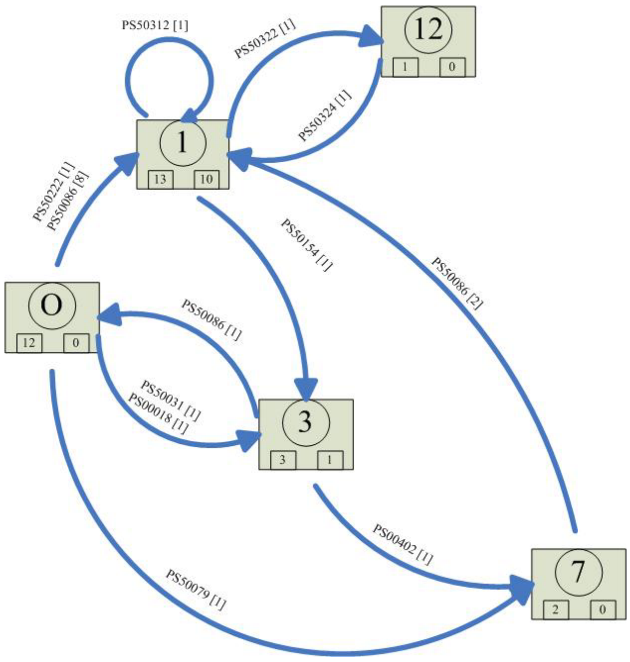

Figure 4. The induced stochastic FSA.

Although it seems that the worst-case complexity of the algorithm is third degree, experimentally, time increases linearly with the size of PTA.

5. CLASSIFICATION RULES INDUCTION

The final step in the proposed methodology is the induction of interesting classification rules. Every rule has three qualitative attributes, i.e. support, confidence and interest. Using the FSA model, constructed by the PTA, these attributes can be defined by calculating the probabilities of symbol appearances in strings. Several algorithms exist for estimating these probabilities and for detecting interesting sequence motifs that occur in large sequential data.

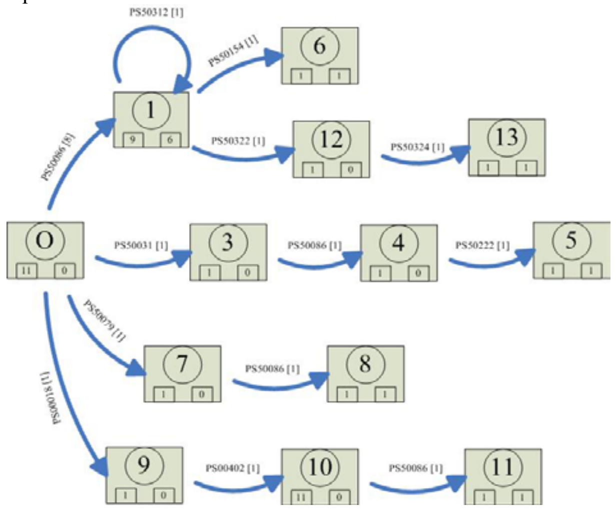

Figure 5. Merging of states 1 and 2, with confidence level 0.8.

However, the datasets in our case are far from sequential. They do not contain any sequence information, i.e. whether a motif comes before or after another. This means that we had to convert existing algorithms to allow for calculation of probabilities in non-sequential data. In order to explain this transformation, first we present the procedure for calculating probabilities of sequence appearances and then the methodology for non-sequential data.

It has been shown that it is possible to compute, from the FSA, the probability of appearance of strings that contain symbol x followed by the symbol y [11]. Since motifs in a protein sequence are not ordered, there is a need to compute the probability of appearance of strings that contain symbols x and y regardless of order. The developed algorithm assigns a probability of appearance to a set of motifs. The probability is calculated recursively using the permutations of the motifs in the set.

Given a set of two motifs, e.g. motifs a and b, the probability of the set appearing in a string can be calculated using the corresponding sequence probabilities:

$$\begin{array}{lrcl} \mathrm{P}(a{+}b) = & +\,\mathrm{P}(a \to b) & + & \mathrm{P}(b \to a) \\ & -\,\mathrm{P}(a \to b \to a) & - & \mathrm{P}(b \to a \to b) \\ & +\,\mathrm{P}(a \to b \to a \to b) & + & \mathrm{P}(b \to a \to b \to a) \\ & +\,\ldots & + & \ldots \end{array}$$

The probability of appearance for a set of motifs can be approximated with an infinite sum, hence a recursive algorithm is proposed. In every recursion, an amount is added to, or subtracted from the total probability, due to the overlapping of probabilities. For example, the string *aba* will be taken under consideration twice, both for the calculation of the probability $\mathrm{P}(a \to b)$ and of the probability $\mathrm{P}(b \to a)$. For this reason, the exceeding amount is subtracted in the next recursion by $\mathrm{P}(a \to b \to a)$. The recursion is terminated when the difference of the computed probabilities before and after the addition or the subtraction is less than a minimum error e.

Generally, for n motifs in the set, the algorithm can be stated as follows:

```
1) Compute all possible permutations of n motifs (n!):

        Perm(j), for j = 0, 1, …, (n! – 1)

2) For every permutation calculate the prefix of length (n – 1) and
the suffix of length 1:

    Prefix(j), Suffix(j), for j = 0, 1, …, (n! – 1)

3) Initialization:

    i = 0:
      Compute the initial value of the probability
                                           using:
      $P = \sum_{j=0}^{n!-1} \left((-1)^i \cdot \mathrm{p}[Perm(j)]\right)$

    i = 1:
      Create new sequences using the old ones and
                     the corresponding prefixes, as:
Perm(j) = Perm(j) + Suffix(j), j = 0, 1, …, (n! – 1)

      Compute new probability value:
      $P' = \sum_{j=0}^{n!-1} \left((-1)^i \cdot \mathrm{p}[Perm(j)]\right)$

4) Recursion:

  While (|P – P´| > e)
    i = i + 1
    P = P´
    If (i mod 2 == 0)
      Perm(j) = Perm(j) + Prefix(j), j = 0, 1, …,
                                            (n! – 1)
    End If
    If (i mod 2 == 1)
      Perm(j) = Perm(j) + Suffix(j), j = 0, 1, …,
                                            (n! – 1)
    End If

    $P' = \sum_{j=0}^{n!-1} \left((-1)^i \cdot \mathrm{p}[Perm(j)]\right)$

  End While
```

In order to extract interesting rules, we exploit the concept of *interest*, defined as:

$$interest(X \to Y) = \frac{\mathrm{p}(X \cup Y)}{\mathrm{p}(X) \cdot \mathrm{p}(Y)} = \frac{supp(X \to Y)}{supp(X) \cdot supp(Y)}$$

Essentially, the interest of the rule expresses the measure of independence between X and Y. As *interest* approaches 1 there is a greater dependence between *X* and *Y*. In order to select interesting rules, we use a threshold value *mininterest*, so that:

$$\left| \frac{p(\mathrm{X} \cup \mathrm{Y})}{p(\mathrm{X}) \cdot p(\mathrm{Y})} - 1 \right| \geq \text{mininterest}$$

The form of the rules that are extracted in the case of protein classification is very specific. The left part is always a combination of motifs, i.e. an unordered set of one or more motifs, while the right part is always a combination of protein families. The classification rules that are extracted have support and confidence that can be defined as:

$$confidence(X \rightarrow Y) = \frac{\begin{array}{c}Number\ of\ proteins\ containing\\ motif\ X\ AND\ belonging\ in\ class\ Y\end{array}}{Number\ of\ proteins\ containing\ X}$$

$$support(X \rightarrow Y) = \frac{\begin{array}{c}Number\ of\ proteins\ containing\\ motif\ X\ OR\ belonging\ in\ class\ Y\end{array}}{Number\ of\ all\ proteins}$$

$$support(X) = \frac{Number\ of\ proteins\ that\ contain\ motif\ X}{Number\ of\ all\ proteins}$$

$$support(Y) = \frac{Number\ of\ proteins\ belonging\ in\ class\ Y}{Number\ of\ all\ proteins}$$

The probabilities computed from the FSA in terms of protein data, can be described as:

$$\mathrm{p}(X, Y) = \frac{\begin{array}{c}Number\ of\ proteins\ that\ contain\\ motif\ X\ and\ belong\ in\ class\ Y\end{array}}{Number\ of\ proteins\ that\ belong\ in\ class\ Y}$$

For example, if an FSA is constructed using 3 different protein classes (Y1, Y2, Y3), it is possible to compute probabilities of the form:

$$\mathrm{p}(X, Y_1 \cup Y_2 \cup Y_3) = \frac{\begin{array}{c}Number\ of\ proteins\ that\ contain\\ motif\ X\ and\ belong\ in\ Y_1 \cup Y_2 \cup Y_3\end{array}}{Number\ of\ proteins\ that\ belong\ in\ Y_1 \cup Y_2 \cup Y_3}$$

for every motif or combination of motifs that appears in the specific protein chains. For the total FSA the corresponding probability is:

$$\mathrm{p}(X) = \frac{Number\ of\ proteins\ that\ contain\ motif\ X}{Number\ of\ all\ proteins}$$

We can observe that using the total FSA, it is possible to directly compute *support*(*X*) for every motif or combination of motifs. The estimation of *support*(*Y*) can be derived during the construction of the corresponding PTA, since *support*(*Y*) is the probability of appearance of a protein class.

The confidence of a rule can be computed using *support*(*X*) and *support*(*Y*), by exploiting a *local* FSA. A local FSA is an FSA that has been constructed using a subset of protein classes. Confidence can then be defined as:

$$confidence\ (X \rightarrow Y) = \frac{support(X,\ local\ FSA) \cdot support(Y)}{support(X,\ total\ FSA)}$$

Finally, the support of a rule can be computed using properties of set theory:

$$\begin{aligned} support(X \rightarrow Y) &= support(X) + support(Y) \\ &\quad - confidence(X \rightarrow Y)\ support(\mathrm{X}) \end{aligned}$$

Thus, in order to extract interesting classification rules with *m* motifs in the left part and *c* classes in the second part, the algorithm can be described as:

1) For every combination *Y* of protein classes (from combinations of 1 to combinations of *c*), compute the probability *support*(*Y*).
2) Create the total FSA (using all classes), and for every combination *X* of motifs (from combinations of 1 to combinations of *m*), compute the probability *support*(*X*).
3) For every combination *Y* of classes (from combinations of 1 to combinations of *c*), create a local FSA. Using this FSA, and for every combination *X* of motifs (from combinations of 1 to combinations of *m*), compute *confidence*($X \rightarrow Y$) and *support*($X \rightarrow Y$). If the interest of the rule is smaller than *mininterest*, reject the rule.

The choice of the values for the variables *c* and *m* was based on statistics from the overall protein chains dataset. By querying the protein database, approximately 94% of the proteins belong in 1 or 2 classes and approximately 96% of the proteins contain 5 motifs at most. Thus, the values $c = 2$ and $m = 5$ seem sufficient.

6. EXPERIMENTS

In order to evaluate the developed algorithm, we have performed comparative tests of two types, and an experiment that included the whole dataset that describes all known proteins. Specifically, first, we compared our algorithm to the results presented in [6], by using constant number of classes and variable ratio of train and test set percentages. Then, a comparative test was conducted against results presented in [12], where the number of classes was the variable and the train test set percentage ratio was constant.

Since we need to construct a generalized FSA, though it accurately describes the data structure, we experimentally concluded that the best value for the parameter confidence of the FSA is 0.8. In this way, every transition between states is performed due to a single motif, while at the same time the FSA has a small number of states. In all experiments we extracted simple classification rules of the form: "*Motif*→ *Class*".

In the first experiment our algorithm was compared against a decision tree algorithm. The protein classes considered in this experiment are: PDOC00064 (a class of oxydoreductases), PDOC00154 (a class of isomerases), PDOC00224 (a class of cytokines and growth factors), PDOC00343 (a class of structural proteins), PDOC00561 (a class of receptors), PDOC00662 (a class of DNA or RNA associated proteins), PDOC00670 (a class of transferases), PDOC00791 (a class of protein secretion and chaperones), and PDOC50007 (a class of hydrolases). For clarity of presentation, the Prosite documentation ID, i.e. the PDOCxxxxx number was used to represent that class. Similarly, the Prosite access number i.e. the PSxxxxx was used to represent that motif pattern or profile. The extracted rules are applied in descending order, based on their interest attribute. Motifs in proteins of the test set are compared against the first rule. If the rule is satisfied, then it is utilized for the protein classification, else the protein is compared against the next rule, until all interesting rules are tested or the protein is classified. In this case, for accuracy of comparison purposes, the test set also included all the proteins in the train set. The results of the experiment are plotted in Figure 6.

We can observe that our algorithm features extremely high percentage of success for classification, even for low train set percentage and having only used the simplest form for classification rules.

The second comparative experiment was conducted against the results from another decision tree algorithm presented in [7]. The ratio for the train and test set percentages was 90/10. Three different sets of classes were used, namely with 10,18,28 classes respectively. In this case the train and test sets are disjoint. Table 2 shows the comparative results for successful classification. Although only simple rules were used, we can observe the efficiency of the algorithm in this case also. The run times for each experiment are approximately 7 min, 18 min and 27 min respectively, using a Pentium 4, 1,51 GHz processor with 512 Mbytes Ram and 768 Mbytes Virtual Ram.

The last experiment contained all known proteins in the dataset and all known protein classes, i.e., approximately 40000 proteins in 1100 different classes. 90% of proteins was used as a train set, while the rest 10% was used as a test set. The percentage of successful classification was 41.4%. This is a very encouraging result, since only simple rules of the form "*Motif* → *Class*" were actually used. Moreover, other methods failed to proccess all 1100 protein classes due to the large amount of protein chains involved. The compactness of our model allows for the full proccessing of the dataset. More motifs in the left part of the rules could probably result in higher percentage of success, though more computational power would be necessary. The run time for this experiment was approximately 7 hours and 34 minutes.

Table 1. Comparison of success percentages for different class sets.

Number of classes	FSA Algorithm	Ref[12]
10	93.713	80.130
18	94.310	84.268
28	75.983	74.490

7. CONCLUSIONS

We presented a graph-based algorithm for the extraction of interesting classification rules. FSAs have been proven to be a compact and efficient solution for the classification problem of proteins based on motifs. Although the classification rules extracted are of a simple form, the algorithm has yielded extremely high percentages of successful classification that

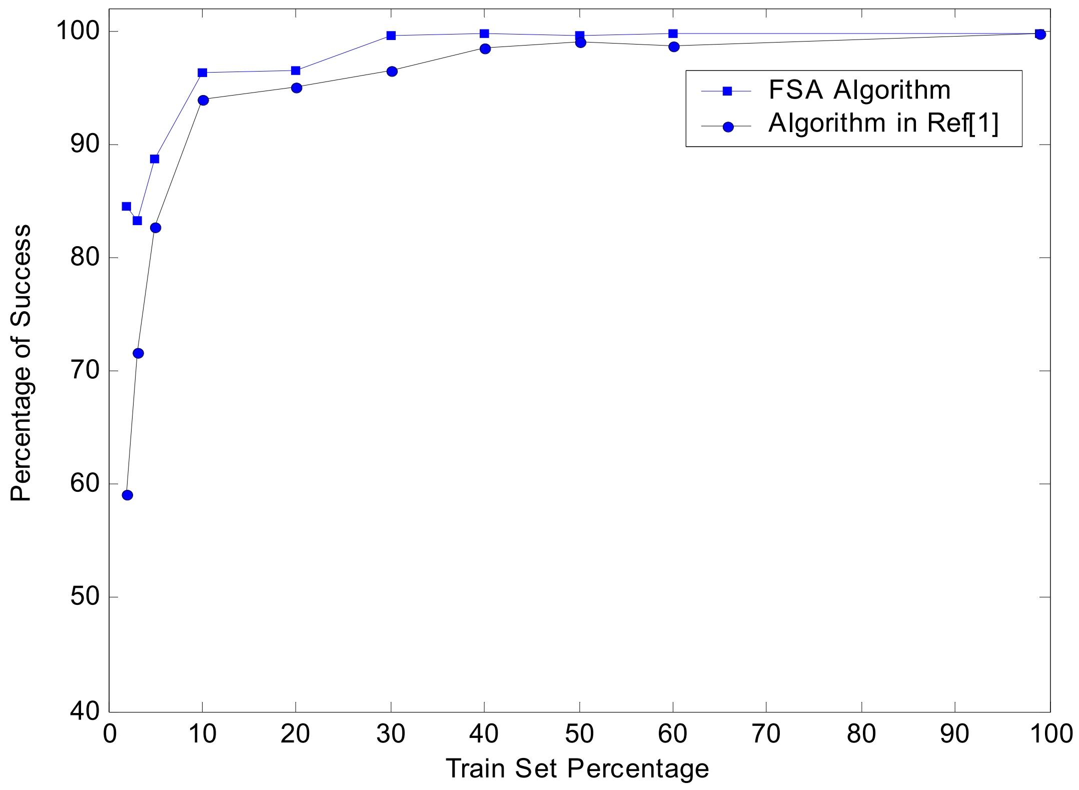

Figure 6. Comparative classification success percentage vs. train set percentage.

reached nearly 100%. Fine-tuning of the algorithm could probably give robust results, that could be more trustworthy for biologists. In the future, we plan to apply the algorithm for more terms in the left part of the rules, in order to increase its efficiency in the whole dataset of proteins and protein classes. Another possible expansion of the algorithm would be to apply it on the classification problem for protein function prediction.

8. REFERENCES

[1] L. Falquet, M. Pagni, P. Bucher, N. Hulo, C.J. Sigrist, K. Hofmann, A. Bairoch, *The PROSITE database, its status in 2002*, Nucleic Acids Res. 30 (2002) 235–238.

[2] A. Bateman, E. Birney, R. Durbin, S.R. Eddy, K.L. Howem, E.L.L. Sonnhammer, *The Pfam protein families database*, Nucleic Acids Res. 28 (2000) 263–266.

[3] T.K. Attwood, M.D.R. Croning, D.R. Flower, A.P. Lewis, J.E. Mabey, P. Scordis, J. Selley, W. Wright, *PRINT-S: the database formerly known as PRINTS*, Nucleic Acids Res. 28 (2000) 225–227.

[4] T. Mitchell, *Machine Learning*, McGraw Hill, New York, 1997.

[5] P.F. Baldi, S. Brunak, *Bioinformatics: The Machine Learning* Approach, The MIT Press, Cambridge, MA, 2001.

[6] D. Wang, X. Wang, V. Honavar, D. Dobbs, Data-driven generation of decision trees for motif-based assignment of protein sequences to functional families, In: *Proceedings of the Atlantic Symposium on Computational Biology*, Genome Information Systems & Technology, 2001.

[7] J.R. Quinlan, *Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1992.

[8] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[9] R. Duad, P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[10] R.C. Carrasco, J. Oncina, Learning stochastic regular grammar by means of a state merging method, In: *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI '94)*, Alicante, Spain, Lecture Notes in Artificial Intelligence, 139 – 152, Springer – Verlag, 1994.

[11] P. Hingston, Using Finite State Automata for Sequence Mining, In: *Proceedings of the 25th Australasian Computer Science Conference, Melbourne*, Australia, 105 – 110, 2001.

[12] G. Hatzidamianos, S. Diplaris, I. Athanasiadis, P.A. Mitkas, GenMiner: a Data Mining Tool for Protein Analysis, In: *Proceedings of the 9th Panhellenic Conference on Informatics*, 2003, Thessaloniki, Greece