A Hybrid Evolutionary Approach for the Protein Classification Problem*

Denise F. Tsunoda[♯†], Heitor S. Lopes[†], and Alex A. Freitas[§]

[†]Bioinformatics Laboratory, Federal University of Technology - Paraná,
Av. 7 de setembro, 3165 80230-901, Curitiba (PR), Brazil

[♯]Department of Information Sciences, Federal University of Paraná, Brazil

[§]Computing Laboratory, University of Kent at Canterbury, UK
dtsunoda@ufpr.br, hslopes@utfpr.edu.br, A.A.Freitas@ukc.ac.uk

Abstract. This paper proposes a hybrid algorithm that combines characteristics of both Genetic Programming (GP) and Genetic Algorithms (GAs), for discovering motifs in proteins and predicting their functional classes, based on the discovered motifs. In this algorithm, individuals are represented as IF-THEN classification rules. The rule antecedent consists of a combination of motifs automatically extracted from protein sequences. The rule consequent consists of the functional class predicted for a protein whose sequence satisfies the combination of motifs in the rule antecedent. The system can be used in two different ways. First, as a stand-alone classification system, where the evolved classification rules are directly used to predict the functional classes of proteins. Second, the system can be used just as an "attribute construction" method, discovering motifs that are given, as predictor attributes, to another classification algorithm. In this usage of the system, a classical decision tree induction algorithm was used as the classifier. The proposed system was evaluated in these two scenarios and compared with another Genetic Algorithm designed specifically for the discovery of motifs – and therefore used only as an attribute construction algorithm. This comparison was performed by mining an enzyme data set extracted from the Protein Data Bank. The best results were obtained when using the proposed hybrid GP/GA as an attribute construction algorithm and performing the classification (using the constructed attributes) with the decision tree induction algorithm.

Key words: Bioinformatics, Protein classification problem, evolutionary computation, genetic programming, genetic algorithm.

1 Introduction

Bioinformatics has become a major field in scientific research due to two main reasons. First, the life sciences community is overwhelmed by the huge amount

* This work was partially supported by the Brazilian National Research Council – CNPq, under research grant no. 309262/2007-0 to H.S. Lopes.

of data available in public databases. Indeed, the size of biological databases is growing at an exponential rate as a consequence of advancements in genome sequencing technology. Second, there has been a significant increase in the amount of available computational power. As a result, some complex problems such as protein folding simulation can now be handled within a reasonable time.

Proteins have several functions within an organism and there are thousands of different types of proteins. They are composed of amino acids linked in linear chains through peptide connections. Proteins are grouped into families according to their biological function, and the classification of proteins is an important task for the molecular biologist.

There are several protein databases freely available in the Internet, and this work is based on the PDB (Protein Data Bank)[3]. This database contains information about the primary, secondary and tertiary structures of more than 56600 proteins (as in march/2009).

The protein classification problem (PCP) is a very important research area in bioinformatics. Basically, the PCP is the discovery of the functional class of an unknown-function protein, by means of analyzing its structure. Most proteins share similar structures (in particular, considering the primary structure), since many of them have a common evolutionary origin [15]. Common structures may be characteristic of a given family of proteins but, on the other hand, unrelated families can also share common structures. This twofold characteristic makes protein classification a difficult problem.

Computer science researchers have been using many different methods to find possible solutions for the PCP, for instance: neural networks [20], clustering algorithms [12], particle swarm optimization [8], genetic algorithms [18] and other data mining algorithms [11],[9].

This paper reports the development and application of a hybrid Genetic Programming (GP)/Genetic Algorithm (GA), specially devised for the automatic discovery of motifs using as input the primary structure information of proteins. The system generates rules based on discovered sequences of amino acids (motifs). These rules cover most proteins of a given class (family) without covering many proteins in other classes. Further, these discovered rules can be used for the characterization of families of proteins as well as for the automatic classification of unknown-function proteins.

2 A Hybrid GP/GA System for Discovering Protein Motifs

The hybrid Genetic Programming/Genetic Algorithm system is detailed here. This system was named HEADMOP (*H*ybrid *E*volutionary *A*lgorithm for the *D*iscovery of *MO*tifs in *P*roteins) and was used for discovering motifs and classification of protein functions.

2.1 Individual Representation

In Genetic Programming (GP) [10], each individual corresponds to a candidate solution to the problem. In HEADMOP, an individual corresponds to a classification rule of the form: *IF (a certain combination of motifs) THEN (predict a certain functional class)*.

The genetic material of an individual consists of a tree containing two kinds of nodes: internal and leaf nodes. Each internal node contains one of the following logical operations: AND, OR, NOT. Each leaf node contains a sequence of amino acids (considering the 20 standard amino acids), also known as a feature or motif. Leaf nodes can contain motifs of different lengths. An example of an individual is shown in Figure 1.

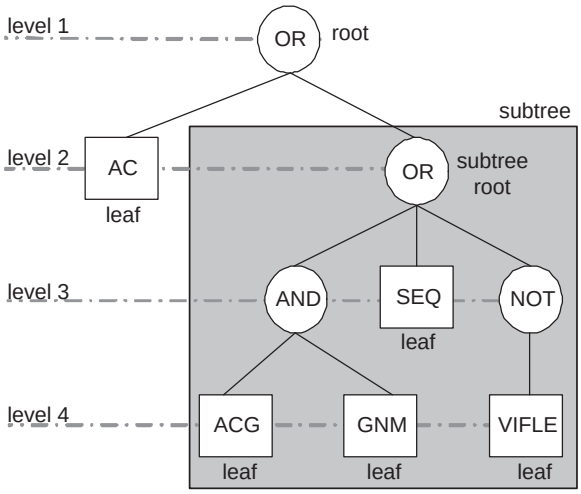

Fig. 1. Example of an individual in the proposed hybrid GP/GA system.

Two categories of genetic operators were defined in HEADMOP: structural and leaf operators. Structural operators work on internal nodes of the tree, modifying the structure of the rule represented by the individual. Leaf operators modify only the sequences of amino acids (the motifs) in leaf nodes. These operators will be discussed in detail in section 2.2.

It should be noted that an individual consists of logical conditions and motifs specifying only the antecedent (the "IF part") of a classification rule. The class predicted by the consequent (the "THEN part") of the rule is not represented in the individual. The class associated with a rule is computed by using a deterministic procedure that assigns the best possible class to the rule, as will be explained later.

Actually, the proposed system can be regarded as a hybrid GP/GA, in the following sense. The individuals' genetic material has internal and leaf nodes (like in GP), but these leafs are encoded as linear genomes (like in GA), where different genomes can have different lengths of amino acid sequences (corresponding to chromosomes with a variable number of genes).

2.2 Selection Method and Genetic Operators

HEADMOP uses stochastic tournament selection, which works as follows. First, k individuals are randomly drawn from the current population, with replacement, where k is a user-specified parameter determined as a percentage of the population size. In this work, k was set to 3% of the population size. Then, these k individuals are put to "play a tournament", in which the probability of an individual to win the tournament is proportional to its fitness value. A copy of the winner of the tournament is then set apart to further undergo the action of the genetic operators.

A) Structural Operators. The proposed system uses the classical sub-tree crossover, where a crossover point is randomly selected in each of the two parent individuals, and then they swap their genetic material rooted at their corresponding crossover points.

Mutation is the other structural operator. It is an asexual operation, involving only one parent individual. This operation consists of randomly selecting one node of the individual and applying a mutation to that point, producing a new child individual. In our system two different structural mutation operations were defined:

1. Sub-tree mutation: this operator replaces the sub-tree rooted at the selected mutation point by a new randomly-generated sub-tree
2. Point mutation: this operator simply replaces the logical function at the selected mutation point (OR, AND or NOT) by another randomly-generated logical function. If an OR or an AND is changed to a NOT, only one of the two sub-trees below the former OR/AND node is maintained (that one with better evaluation) – the other is simply deleted to maintain the individual's integrity. If a NOT is changed to an OR or an AND a sub-tree is randomly-generated and inserted as the other sub-tree below the new OR/AND node, also to maintain the individual's integrity.

Both structural crossover and mutation operators can be used as "hill-climbing operators", in the following sense. In the case of the mutation operator, immediately after the creation of a new child individual, its fitness is computed. If the child's fitness is lower than the parent's fitness, the parent (rather than the child) is copied to the next generation. In the case of the crossover operator, the same holds. The individual with best fitness among parents and children is copied to the next generation. The hill-climbing behavior of the structural operators is probabilistic, controlled by a user-defined parameter named "probability of hill climbing" (see section 3.2).

B) Leaf Operators The conventional one-point crossover in genetic algorithms was originally designed for a fixed-length chromosome representation [6]. Hence, it cannot be applied directly to the leaf nodes of our approach, where a motif is represented by a variable-length sequence of amino acids. Therefore, we adapted

the conventional one-point crossover to a variable-length representation, as follows. The crossover point (which is still randomly generated) indicates the percentage of the amino acid sequence of the leaf node in each parent where the swapping of amino acids ("genes") starts. The percentage (relative position) is the same for both leaf nodes (i.e., one leaf node of each parent GP individual), but the actual (absolute) position where the amino acid swapping starts can be different, since the two leaf nodes can have different numbers of amino acids.

This work introduces four different mutation operators devised for dealing with the variable-length sequence of amino acids contained in a leaf node of an individual, as follows.

1. Addition to the Left (AL): a letter (representing an amino acid) is randomly generated and inserted into the leftmost end of the sequence of amino acids;
2. Addition to the Right (AR): analogous to AL, except that the new amino acid is inserted into the rightmost end of the sequence of amino acids;
3. Multiple Mutations (MM): all the amino acids starting from a randomly-generated position up to the end of the sequence are replaced by other randomly-generated amino acids. The starting position can be any in the sequence, except the first and the last ones.
4. Removal (RM): the amino acid in a randomly chosen position is removed from the sequence. The application of this operator is subject to the constraint that after removal of an amino acid the remaining sequence still has at least three amino acids. If this condition is not met then this operator is not applied, and another mutation operator is applied instead.

2.3 Fitness Function

Since the goal is to maximize classification accuracy, the quality of a rule is determined by its ability in discriminating proteins of different classes. That is, ideally a rule should cover all the proteins of a given class and none of all other classes. The fitness function was designed to take this basic principle into account. First, for each class i ($i = 1, \ldots, n$) we compute a measure of coverage of the rule for that class, called F_i. This measure is defined in the range [0..1] and corresponds to the number of proteins of the i-th class covered by the rule, divided by the total number of proteins belonging to the i-th class. A protein is said to be covered by a rule if and only if the protein's primary sequence of amino acids satisfies the conditions of the rule antecedent. Next, for each class i, a measure of the ability of the rule to discriminate between class i and the other classes is computed. This measure is denoted $Disc_i$ and it is defined by equation 1:

$$Disc_i = F_i \cdot \left[1 - \sum_{j=1}^{n} \left(\frac{F_{j,j \neq i}}{n-1} \right) \right] \quad (1)$$

2.4 Result Designation

As explained earlier, each individual represents a rule that is associated with a given class of proteins. Therefore, it is not enough to return as a solution only the

best rule found throughout the evolutionary process – as usual in conventional evolutionary algorithms. Indeed, it is necessary to return a set of rules, since different rules can predict different classes. Hence, the result provided by our system consists of the best M rules for each class found throughout the evolutionary process, where M is a user-defined parameter. This was implemented by elitism, in such a way that, at every generation, the M best individuals of each class are preserved in the population. Recall that the quality of a rule (Equation 1 is based on its coverage.

The set of rules returned by HEADMOP was used for classification in two ways. First, each rule (individual) was used as a complete classification rule. In this case, when a new protein is to be classified, the system counts how many rules associated with each class cover that protein. The protein is assigned to the class with the highest value of that count, i.e., the class with the largest number of rules covering the protein. This is effectively a majority voting strategy, where each class has as many votes as the number of rules that predict that class and cover the protein.

In the second approach, each returned rule is interpreted as a binary predictor attribute, which takes on the value true if a protein satisfies the antecedent of the rule and takes on the value false otherwise. Note that, in this case, the rule consequent is effectively ignored, i.e., the new predictor attribute is constructed based only on the information derived from the rule antecedent - the motifs in the leaf nodes and the logical conditions in the internal nodes. The motivation for this approach – which is more complex than the previously-discussed first approach – is that this new, higher-level data set can be given to any classification algorithm. In this case the system is effectively acting as a sophisticated "attribute construction" (feature discovery) algorithm and it delegates the task of actually building the classifier to a another classification algorithm.

In order to implement this second approach, the chosen classification algorithm was J48, which is a Java implementation of the very well-known, and widely used, C4.5 algorithm [16]. J48 is available as part of a data mining tool named WEKA [22], which has the advantage of being freely available and widely used [1]. The choice of J48 was further motivated by the fact that this algorithm produces a decision tree, a classification model that tends to be comprehensible to the user, allowing him/her to interpret discovered knowledge. This is important not only in data mining in general, but also in Bioinformatics applications [14], [4], [17], where the goal is to give the user some new insight about the relationships that hold in the data.

3 Computational Experiments and Results

3.1 Data Set Used in the Experiments

The data set used in the experiments was extracted from the PDB. First, records which had an EC number were set apart and, from these, a random subset was

[1] http://www.cs.waikato.ac.nz/~ml/weka/

extracted. Proteins having an EC number are enzymes and the code is provided by IUBMB (International Union of Biochemistry and Molecular Biology). From a data mining viewpoint, each EC number corresponds to a class, i.e., a specific protein function. More precisely, the EC number consists of four digits, where each pair of adjacent digits is separated by a dot ("."), and it specifies the chemical reaction catalyzed by the corresponding enzyme. For instance, the enzyme Alcohol dehydrogenase has the number EC.1.1.1.1.

Note that this is a four-level hierarchical classification, so that the first digit represents the most general class and the last digit the most specific subclass. In this work we report results of enzyme classification at the four levels, one-level-at-a-time. That is, first an experiment is carried out to classify enzymes at the first level (i.e. predict the first EC code digit); then a separate, independent experiment is carried out to classify enzymes at the second level; and so on, until experiment four that predicts the fourth EC code digit. Ultimately, these experiments aim at predicting the functional class of enzymes.

It should be noted that these experiments are more extensive than other experiments with enzyme classification reported in the literature, where sometimes just the first EC code digit is predicted – see e.g. [20], [2] and [1]. A direct comparison between different methods is not possible due to the differences in the data sets used and in the methodology for computing results. Also, presumably, the main reason why the literature sometimes focuses on just the first-level classification is that, as we consider deeper levels of classification, the number of examples per class becomes smaller and smaller, so that the classification problem becomes harder and harder. Indeed, in the enzyme data set, many classes at the third and fourth level have so few examples (less than 10) that they can hardly be predicted with a reasonable accuracy. This is not a limitation associated with any classification algorithm by itself, it is simply a limitation associated with the data being mined. Therefore, as part of our data preparation procedure, we have retrieved from PDB only the enzymes belonging to subclasses with at least 10 elements. After this simple filtering, the total number of enzymes retrieved from PDB and the corresponding number of classes are shown in Table 1 for all four levels of classification.

Table 1. Characteristics of the enzyme dataset used in the experiments.

Enzyme class level	Number of enzymes	Number of classes
EC.X	11,493	6
EC.X.X	11,455	56
EC.X.X.X	10,094	178
EC.X.X.X.X	7,725	1036

3.2 Running Parameters

Preliminary experiments were done to adjust the parameters of the algorithm. These experiments were done evolving motifs with the primary structure of pro-

teins. As a result of the preliminary experiments, parameters were adjusted as follows: population size = 500; number of generations = 20; tournament size = 3% (of the population size); probability of structural crossover = 60%; probability of structural mutation = 60%; probability of leaf crossover = 20%; probability of leaf mutation = 80%; probability of hill climbing = 40%; number of discovered rules per class (M) = 10; stopping criterion = maximum number of generations.

From now on these parameter values will be referred to as default values, and they were used in all other experiments reported in this paper. It should be stressed that these preliminary experiments were not exhaustive and it is possible that other set of running parameters could lead to better performance. Actually, in general, adjusting running parameters in evolutionary computation systems is still an open issue. As a consequence of the lack of a widely accepted methodology for adjusting parameters, current research is pointing to self-adapting strategies [13]. Possibly, this issue will be addressed in future developments.

3.3 Comparative Results for Motif Discovery

In order to evaluate the performance of HEADMOP, we have compared it with a Genetic Algorithm (GA) that was also designed for motif discovery. This GA was previously described in the literature, as GAMDI [18], and an improved version as GAMBIT [19] The main differences between GAMDI and HEADMOP, are as follows. In GAMDI each individual corresponds to a single motif, whereas in HEADMOP each individual corresponds to a rule consisting of logical operators that combine several motifs (one motif for each leaf node of the individual). Also, the motifs discovered by GAMDI have to be given, as predictor attributes, to another classification algorithm which, actually, will discover classification rules. By contrast, since each individual in HEADMOP already corresponds to a potentially complex rule, each rule it discovers can be used on its own, without the need for a classification algorithm. Notwithstanding, it is also possible to use each of the individuals returned by this system as a single predictor attribute to be given to a classification algorithm. This approach is particularly interesting to implement a fair comparison between the two evolutionary algorithms, since the motifs discovered by both are used by the same classification algorithm. Hence, we report two types of results for HEADMOP:

1. Results using the system as a stand-alone data mining algorithm – in which the system both discovers motif-based rules and uses those rules directly for enzyme classification.
2. Results when the system is used to discover motifs that are then used by another separate classification algorithm as predictor attributes.

Results of both experiments are compared with the results of GAMDI discovering motifs that are then used, as predictor attributes, by a separate classification algorithm. In order to make the comparison between HEADMOP and GAMDI as fair as possible, both the previously-mentioned experiment (2) and the experiments with GAMDI used the same classification algorithm, namely J48 [22], as explained earlier.

Many works in current literature deals with a two-classes problem, rather than a multiclass problem (such as the one approached in this work). Viewing a multiclass problem as a two-classes one, the cases of a given class could be considered as "positive" and the cases of all remaining classes together are considered "negative". Table 2 shows the accuracy rates obtained by the algorithms for each class level of the EC code hierarchical classification. In this table, numbers after the "±" symbol denote the corresponding standard deviations. All accuracy rates in this paper refer to the average predictive accuracy on a test set unseen during training, as measured by a 5-fold cross-validation procedure [22]. The test set was randomly chosen with 1/3 of the cases of the data set shown in Table 1, maintaining the proportion of classes. We choose to present only the accuracy rate as a measure of quality in order to make possible the comparison with other published work.

Table 2. Classification accuracy rates (%) on the test set.

Enzyme class level	GAMDI	HEADMOP	HEADMOP+J48
EC.X	83.32±1.34	78.25±0.65	87.82±1.31
EC.X.X	79.64±0.43	77.19±0.98	83.33±0.54
EC.X.X.X	72.54±1.41	78.79±0.74	79.10±0.56
EC.X.X.X.X	75.85±5.26	79.19±0.89	83.13±0.49

As shown in Table 2, using the HEADMOP system as a stand-alone data mining algorithm mixed results were obtained. That is, HEADMOP achieved classification accuracies lower than the GAMDI algorithm at the first and second class levels, but higher than GAMDI at the third and fourth class levels. The differences in accuracies are significant at the first, second and third class levels, considering that the corresponding standard deviation intervals do not overlap. The difference in accuracy between GAMDI and HEADMOP alone is not significant, however, at the fourth level, where the standard deviation intervals overlap.

A better result was obtained when HEADMOP was used only to produce the predictor attributes used by J48, as explained before. In this case HEADMOP+J48 outperformed both GAMDI and the use of HEADMOP alone for all class levels. Additionally, the differences in accuracy between HEADMOP+J48 and GAMDI were significant at all class levels. These results suggest that HEADMOP is actually discovering good motifs, encoded in the leaf nodes of individuals. However, the logical combination of the motifs represented in the leaf nodes – performed by the logical operations in the internal nodes of individuals – still leaves room for improvement. Indeed, the use of J48 to combine the motifs represented by HEADMOP leaf nodes seems a good way to implement such an improvement, since the accuracies of HEADMOP+J48 are clearly higher than the accuracies of HEADMOP alone in general. This result is consistent with the fact that J48 is the product of several decades of research in machine learning,

as mentioned earlier. Anyway, HEADMOP still has the important merit of discovering high-quality motifs, without which J48 would be helpless – since, of course, J48 cannot discover protein motifs.

3.4 Comparative Results for Processing Time.

It is also interesting to compare the computational time taken by GAMDI and HEADMOP. The result of this comparison is presented in Table 3, in the format *hours* : *minutes*. All experiments were run in PC desktops with AMD Athlon XP 2.4 processors.

Table 3. Total running time of the algorithms(hr:min).

Enzyme class level	GAMDI	HEADMOP
EC.X	03:42	0:39
EC.X.X	10:04	0:30
EC.X.X.X	16:06	1:04
EC.X.X.X.X	19:34	0:57

As can be observed in Table 3, HEADMOP is considerably faster than GAMDI. In the case of the enzyme data set used in this project, the differences in the processing time of the algorithms is not crucial, since the longest running time, taken by GAMDI, was shorter than one day. However, in much larger data sets the long processing time taken by GAMDI would probably be a serious limitation to the use of that algorithm, whilst the hybrid GP/GA system proposed in this paper seems much more scalable to larger data sets. Further research will also focus on the parallelization of these algorithms.

4 Conclusions

We have proposed a hybrid Genetic Programming/Genetic Algorithm system for rule discovery, aiming at the automatic functional classification of proteins with unknown function. The system, named HEADMOP, was evaluated using an enzyme data set extracted from the Protein Data Bank. The solution returned by the proposed system consists of a set of rules (individuals), each of them using logical operators to combine a set of motifs and predicting a certain class for all enzymes satisfying the rule represented by the individual.

More precisely, each solution returned by HEADMOP consists of two major components evolved by the algorithm, namely, a set of protein motifs in the leaf nodes and a set of logical operators in the internal nodes of the individual. Hence, it is desirable to do controlled experiments evaluating the effectiveness of each these two solution components separately. In this spirit, we evaluated the results of the system when it was used in two different ways. In the first approach

HEADMOP was used as a stand-alone classification algorithm, so that both kinds of evolved solution components (the set of motifs and the logical operators combining the motifs) were used to predict the class of an enzyme. In the second approach, HEADMOP was used only as an "attribute construction" algorithm, in the sense that only the motifs discovered by the system (and not the logical operators) were used. In this case the discovered motifs were used as attributes by J48, a standard classification algorithm based on decision trees.

The results of both approaches for using the solution returned by HEADMOP were compared with the results of a previous work using GA (GAMDI) that was also especially designed for the discovery of protein motifs. Unlike HEADMOP, GAMDI evolves only motifs, and not logical operators combining motifs. Hence, the motifs discovered by GAMDI have to be given (as attributes) to a classification algorithm. Again, J48 was used, to make the comparison between HEADMOP and GAMDI as fair as possible.

The performance measure used in this comparison was classification accuracy on a test set separated from the training set, as usual in the data mining literature. The two main findings from these experiments are as follows. First, there was no clear winner in the comparison between HEADMOP used as a stand-alone classification algorithm and J48 using motifs discovered by the GAMDI. Second, the results of J48 using the motifs discovered by HEADMOP were clearly better than the results of J48 using the motifs discovered by the former GA-based system. Hence, the general conclusion from these results is that the hybrid GP/GA system is evolving good sets of motifs in the leaf nodes of the individuals, but the combination of those motifs via the logical operators in the internal nodes is not fully effective – suggesting more research in this topic in the future.

It is intended to develop a more sophisticated version of HEADMOP where the motifs use information about the secondary structure of proteins. We believe that incorporating more information in the motifs can improve the classification accuracy of the system, especially covering atypical proteins, considered as small disjuncts [21] in the classification task. Authors intend to make such version freely available in the internet so as to foster further research in the area.

Results, in general, encourages the continuity of the research. Future work will include the use of HEADMOP with other categories of proteins, such as the globins, or else, the use the obtained classification rules to determine the functional class of recently discovered proteins, not yet classified. Overall, we believe that this work can be useful not only for biologists, but also for those working in this important field of Bioinformatics.

References

1. Arakaki, A.K., Zhang, Y., Skolnick, J.: Large-scale assessment of the utility of low-resolution protein structures for biochemical function assignment. *Bioinformatics* **20** (2004) 1087–1096
2. Ben-Hur, A., Brutlag, D.: Remote homology detection: a motif based approach. *Bioinformatics* **19** (2003) i26–i33

3. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. *Nucleic Acids Research* **28** (2000) 235–242
4. Clare, A., King, R.D.: Machine learning of functional class from phenotype data. *Bioinformatics* **18** (2002) 160–166
5. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview. In: Fayyad, U.M. et al. (Eds.) *Advances in Knowledge Discovery and Data Mining* (1996) 1–34
6. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading (1989)
7. Han, J., Kamber, M.: *Data Mining: Concepts and Techniques.* Morgan Kaufmann, San Mateo (2001)
8. Holden, N., Freitas, A.A: Hierarchical classification of protein function with ensembles of rules and particle swarm optimisation. *Soft Computing* **13** (2009) 259–272
9. King, R.D., Karwath, A., Clare, A., Dehaspe, L.: The Utility of different representations of protein sequence for predicting functional class. *Bioinformatics* **17** (2001) 445–454
10. Koza, J.R.: *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge (1992)
11. Kretschmann, K., Fleischmann, W., Apweiler, R.: Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on Swiss-Prot. *Bioinformatics* **17** (2001) 920–926
12. Manning, A.M., Brass, A., Goble, C.A., Keane, J.A.: Clustering techniques in biological sequence analysis. In: *Proceedings of the* 1^{st} *European Symposium on Principles of Data Mining and Knowledge Discovery* (1997) 315–322
13. Maruo, M.H., Lopes, H.S., Delgado, M.R.B.S.: Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. *Lecture Notes in Computer Science* **3448** (2005) 154–165
14. Mirkin, B., Ritter, O.: A Feature-based approach to discrimination and prediction of protein folding groups. In: Suhai, S. (Ed.), *Genomics and Proteomics: Functional and Computational Aspects.* Kluwer, Doordrecht (2000) 157–177
15. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* **247** (1995) 536–540
16. Quinlan, J.R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Francisco (1993)
17. Sebban, M., Mokrousov, I., Rastogi, N., Sola, C.: A data mining approach to spacer oligonucleotide typing of mycobacterium tuberculosis. *Bioinformatics* **18** (2002) 235–243
18. Tsunoda, D.F., Lopes, H.S.: Automatic motif discovery in an enzyme database using a genetic algorithm-based approach. *Soft Computing* **10** (2006) 325–330
19. Tsunoda, D.F., Lopes, H.S., Freitas, A.A.: An evolutionary approach for motif discovery and transmembrane protein classification. *Lecture Notes in Computer Science* **3449** (2005) 105–114
20. Weinert, W.R., Lopes, H.S.: Neural networks for protein classification. *Applied Bioinformatics* **3** (2004) 41–48
21. Weiss, G.M.: Learning with rare cases and small disjuncts. In: *Proc. of Twelfth International Conference on Machine Learning* (1995) 558–565
22. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 2^{nd} edition. Morgan Kaufmann, San Francisco (2005)