Improving the Performance of Hierarchical Classification with Swarm Intelligence

Nicholas Holden and Alex A. Freitas

Computing Laboratory, University of Kent,
Canterbury, CT2 7NF, UK
nickpholden@gmail.com, A.A.Freitas@kent.ac.uk

Abstract. In this paper we propose a new method to improve the performance of hierarchical classification. We use a swarm intelligence algorithm to select the type of classification algorithm to be used at each "classifier node" in a classifier tree. These classifier nodes are used in a top-down divide and conquer fashion to classify the examples from hierarchical data sets. In this paper we propose a swarm intelligence based approach which attempts to mitigate a major drawback with a recently proposed local search-based, greedy algorithm. Our swarm intelligence based approach is able to take into account classifier interactions whereas the greedy algorithm is not. We evaluate our proposed method against the greedy method in four challenging bioinformatics data sets and find that, overall, there is a significant increase in performance.

Keywords: Particle Swarm Optimisation, Ant Colony Optimisation, Data Mining, Hierarchical Classification, Protein Function Prediction.

1 Introduction

Hierarchical classification is a challenging area of data mining. In hierarchical classification the classes are arranged in a hierarchical structure, typically a tree or a DAG (Directed Acyclic Graph). In this paper we consider classes arranged in a tree structure where each node (class) has only one parent – with the exception of the root of the tree, which does not have any parent and does not correspond to any class. Hierarchical class datasets present two main new challenges when compared to flat class datasets. Firstly, many (depending on the class depth) more classes must be assigned to the examples. Secondly, the prediction of a class becomes increasingly difficult as deeper class levels are considered, due to the smaller number of examples per class.

In this paper we address the problem of hierarchical protein function prediction, a very active research topic in bioinformatics. The prediction of protein function is one of the most important challenges faced by biologists in the current "post-genome" era. The challenge lies in the fact that the number of proteins discovered each year is growing at a near exponential rate [1] (with the vast majority of them having unknown function) and advances in the understanding of protein function are critical for more effective diagnosis and treatment of disease, also helping in the design of more effective medical drugs etc.

E. Marchiori and J.H. Moore (Eds.): EvoBIO 2008, LNCS 4973, pp. 48–60, 2008.

In this paper we propose a new method to increase the accuracy of classification when using the top-down divide and conquer (TDDC) approach for hierarchical classification (as described in section 2). The new method is based on a swarm intelligence algorithm, more precisely a hybrid particle swarm optimisation/ant colony optimisation (PSO/ACO) algorithm.

The remainder of this paper is organised as follows: Section 2 introduces hierarchical classification. Section 3 describes an approach proposed by Secker at al. [3] for improving hierarchical classification accuracy and critiques it. Section 4 describes the proposed novel method for hierarchical classification using a swarm intelligence (PSO/ACO) algorithm. Section 5 describes experimental set-up. Section 6 describes the experimental data from four challenging "real-world" biological data sets and section 7 draws conclusions based on the results of the experiments and suggests future research directions.

2 A Brief Review of Hierarchical Classification

This paper focuses on hierarchical classification problems where the classes to be predicted are organized in the form of a tree, hereafter referred to as a class tree populated by class nodes. An example of a hierarchical classification problem might be the prediction of what species and then breed a pet is. In the first case we wish to known whether the given animal is of the class node (species) dog or cat, and in the second case if the animal is of the class node (breed) Burmese, British Blue, Jack Russell or Golden Retriever. In this paper the species would be considered the first class level and the breed the second class level. The TDDC approach is based on the principle that only sibling class nodes need be considered at any point in the hierarchical tree. So at the first set of sibling class nodes (cat or dog) if we decide cat, then at the second set of class nodes we must only decide between the sibling class nodes Burmese or British Blue. Notice that this has a major drawback, which is that if the pet is in fact a dog we are guaranteed to guess the breed wrong if we predict cat at the first class level.

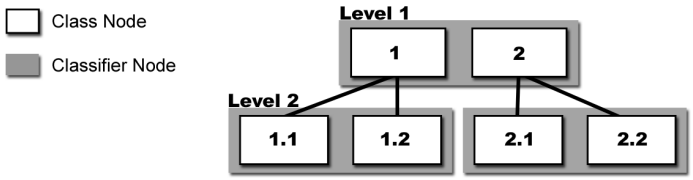

Fig. 1. A Hierarchical classification problem using the TDDC approach

This top-down approach has the important advantage of using information associated with higher-level classes in order to guide the prediction of lower-level classes. This has shown to increase accuracy over other basic approaches [4]. For instance, (from Figure 1), if class 1.X (where X denotes any digit) is predicted at the first level and that class node only has the child nodes 1.1 and 1.2, only these two class nodes should be considered and not the children belonging to node 2.X, 2.1 and 2.2. In Figure 1 the classifier nodes are shown by the grey boxes. There would be classifiers to distinguish between classes 1.X and 2.X, 1.1 and 1.2 etc. It is important to distinguish between two conceptually distinct – though clearly related – trees, namely a

class tree and a classification tree. In the class tree every node represents a class (to be predicted by a classifier). By contrast, in the TDDC tree each node represents a classifier that discriminates between sibling classes. The nodes of a classifier tree are hereafter referred to as classifier nodes. The terms classifier tree and TDDC tree are used interchangeably in this paper.

3 The Greedy Selective Top Down Divide and Conquer Approach

In the conventional top-down approach for hierarchical classification, in general, the same classification algorithm is used for each classifier node. Intuitively, this is a suboptimal approach because each classifier node is associated with a different classification problem – more precisely, a different training set, associated with a different set of classes to be predicted. This suggests that the predictive accuracy of the classifier tree can be improved by selecting, at each classifier node, the classification algorithm with best performance in the classification problem associated with each node, out of a predefined list of candidate classification algorithms. Indeed it was found in [3] by Secker et al. that by varying the classification algorithm at each classifier node in the Top-Down Divide and Conquer (TDDC) tree, classification accuracy could, in general, be somewhat improved.

In Secker's work the training set at each classifier node is divided into two non overlapping sub sets, a building set – used to train the classification algorithms – and a separate validation set – which is used to assess the predictive accuracy of the models constructed by the classification algorithms. At every classifier node in the TDDC tree, multiple classifiers are built using the building set, each using a different classification algorithm. The classification accuracy of each of these classifiers is measured using the validation set at each classifier node, and then the best classifier (according to classification accuracy in the validation set) is chosen. This process is repeated at each classifier node to select a set of classifiers to populate the TDDC classification tree, which is then used to classify the test instances (unseen during training). A simple example of a classification tree constructed by this method, showing a different classifier chosen at each node, is shown in Figure 2.

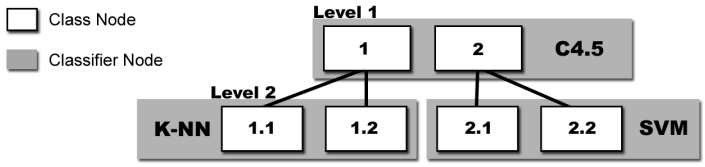

Fig. 2. A TDDC tree using classification algorithm selection

In this way Secker's work uses a greedy selective approach to try and maximise classification accuracy. It is described as greedy because, when it selects a classifier at each classifier node, it maximises accuracy only in the current classifier node, using local data. Therefore, the greedy selective approach ignores the effect of this local selection of a classifier on the entire classifier tree. In other words, this procedure is "short sighted", and so it does not consider the interaction between classifiers at different classifier nodes.

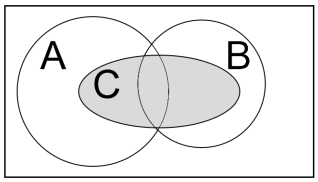

Fig. 3. Classifier interaction scenario where $|B \cap C| > |A \cap C|$

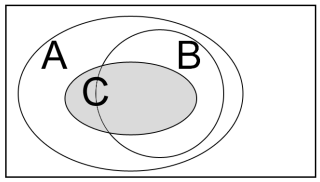

Fig. 4. Classifier interaction scenario where $|B \cap C| < |A \cap C|$

Figures 3 and 4 show two possible scenarios demonstrating interactions between classifiers at different classifier nodes during classifier evaluation. A and B are the two possible parent classifiers trying to discriminate between classes 1 and 2. C is the child classifier that attempts to discriminate between classes 1.1 and 1.2 – as shown in Figure 5. Figures 3 and 4 show the sets of correctly classified examples for each classifier in the TDDC tree. Notice that $C \subseteq A \cup B$ for the three classifiers A, B and C. This is due to the fact that in the standard TDDC tree once a misclassification has been made, by classifiers A or B at the first classifier node, it cannot be rectified by C at the child classifier node.

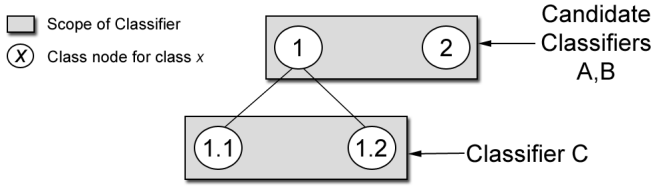

Fig. 5. A class tree used to illustrate the discussion on classifier interaction

As mentioned earlier, the greedy approach chooses the best classifier at each node according to the classification accuracy, in the validation set, at that node. In the scenarios shown in both Figures 3 and 4 classifier A would be chosen to discriminate between classes 1 and 2, as it is more accurate when compared to classifier B, i.e. its circle has a bigger area, denoting a greater number of correctly classified examples. Let us now discuss how appropriate the choice of classifier A (made by the greedy approach) is in each of the different scenarios shown in Figures 3 and 4, taking into account the interactions between classifiers A and C, and between B and C, in the context of the class tree shown in Figure 5.

Recall that in the TDDC approach an example is correctly assigned to class 1.1 or 1.2 if and only if the two following events occur: the example is correctly classified by the root classifier (A or B); and the example is correctly classified by classifier C. Therefore, the individual accuracy of each classifier is not necessarily the most important factor when selecting a candidate classifier; rather it is the number of examples correctly classified by *both* the parent and child classifiers (the intersection between their sets of correctly classified examples). In the case of Figure 5, in order to maximise the classification accuracy at the leaf class nodes 1.1 and 1.2, if $|A \cap C| > |B \cap C|$

then classifier A should be chosen; if it is not, B should be chosen. For this reason, the greedy approach produces a good selection in the case of Figure 4, where $|A \cap C| > |B \cap C|$. However, the greedy approach would not produce an optimal selection in the case of Figure 3. This is due to the fact that although A has a greater area (higher accuracy) in Figure 3, $|B \cap C| > |A \cap C|$.

4 Global Search-Based Classifier Selection with a Particle Swarm Optimisation/Ant Colony Optimisation Algorithm

Given the discussion in the previous section it is quite clear that there is a potential to improve the classification accuracy of the entire classifier tree by using a more "intelligent" classifier selector – a classifier selector that (unlike the greedy one) takes into account interaction among classifiers at different classifier nodes. As there is an obvious objective function to be optimised – the classification accuracy of the entire TDDC tree on the validation set – and also a collection of elements whose optimal combination has to be found – the type of classifier at each classifier node, it seems appropriate to use a combinatorial optimisation algorithm.

We propose to optimise the selection of a classifier at each classifier node with a PSO/ACO algorithm, adapted from the PSO/ACO algorithm described in [4] [5]. The choice of this algorithm was motivated by the following factors. Firstly PSO/ACO has been shown to be an effective classification-rule discovery algorithm [4] [5] across a wide variety of data sets involving mainly nominal attributes. Secondly, the PSO/ACO algorithm can be naturally adapted to be used as a classifier selector, where instead of finding a good combination of attribute-values for a rule, it finds good combinations of classifiers for all the nodes of the classifier tree. This is because a combination of classifiers is specified by a set of nominal values (types of classification algorithms). Due to size restrictions this section assumes the reader is familiar with standard PSO [6] and ACO algorithms [7].

A hybrid (PSO/ACO) method was developed to discover classification rules from categorical (nominal) data [4] [5]. In essence, this algorithm works with a population of particles. Each containing multiple pheromone vectors – each pheromone vector is used to probabilistically decide which value of a nominal attribute is best in each dimension of the problem's search space. In the original PSO/ACO for discovering classification rules these dimensions correspond to predictor attributes of the data being mined, so there is one pheromone vector for each nominal attribute. The entries in each individual pheromone vector correspond to possible values the attribute can take, and each pheromone value denotes the "desirability" of including the corresponding attribute value in a rule condition. We now describe in detail how this algorithm was adapted to act as a classifier selector, rather than discovering classification rules.

To optimise the classifier selection at each classifier node the problem must be reduced to a set of dimensions and possible values in each dimension. Hence, in the proposed PSO/ACO for classifier selection each decoded particle (candidate solution) consists of a vector with n components (dimensions), as follows:

$$\textit{Decoded Particle} = w_1, w_2, \ldots, w_n$$

Where w_d (d=1,..,n) is the classifier selected at the dth classifier node in the TDDC tree and n is the number of classifier nodes in the tree. Each w_d can take one of the

nominal (classifier ids) values $c_1,..c_k$ where k is the number of different candidate classifiers at each node.

It must also be possible to assess how good an individual solution created from an individual particle is. To do this the validation set is classified by the TDDC tree composed of the classifiers specified by the particle, and that tree's average classification accuracy (the mean of the accuracy from each class level) on the validation set is taken. The mean classification accuracy across all the class levels is used as the "fitness" (evaluation) function for evaluating each particle's quality.

Note that the only increase in computational time for this approach (over the greedy selective approach) is in the time spent classifying examples at each fitness evaluation. The classifiers are trained using the same data at each fitness evaluation and so can be cached and reused without the need for retraining.

```
Initialize population
REPEAT for MaxInterations
   FOR every particle P
      /* Classifier Selection */
      FOR every dimension w_d in P
         Use fitness proportional selection on pheromone vector
         corresponding to w_d to choose which state (classifier
         id) c_1,..c_k should be chosen for this w_d
      END FOR
      Construct a classifier tree by using the classifiers se-
      lected from the particle's pheromone vectors
      Calculate fitness F of this set of classifiers w_1,..w_n
      /* Set the past best position */
      IF F > P's best past combination's (P_b) fitness F_b
         F_b= F
         P_b = the current combination of classifiers w_1,..w_n
      END IF
   END FOR
   FOR every particle P
      Find P's best Neighbour Particle N according to each
      neighbour's best fitness (F_b)
      FOR every dimension w_d in P
         /* Pheromone updating procedure */
         f = N's best fitness F_b
         y = N's best state P_b in dimension d
         /* Add an amount of pheromone proportional to f to the
         pheromone entry for particle P corresponding to y (the
         best position held by P's best Neighbour) */
         τ_pdy= τ_pdy + (f × á)
         Normalize τ_pd
      END FOR
   END FOR
END REPEAT
```

Pseudocode 1. The Hybrid PSO/ACO Algorithm for Classifier Selection

Pseudocode 1 shows the hybrid PSO/ACO algorithm for classifier selection. At each iteration each pheromone vector for each particle produces a state in a probabilistic manner. That is, the probability of choosing a given classifier ($c_1,..c_k$) for a given classifier node ($w_1,..w_n$) is proportional to the amount of pheromone (a number between 0 and 1) in the corresponding entry in the corresponding pheromone vector

(τ_{pd} is the pheromone vector corresponding to particle P and classifier node d), see Figure 6. More precisely, the selection of a classifier at each classifier node is implemented by a fitness proportional (roulette-wheel) selection mechanism [8].

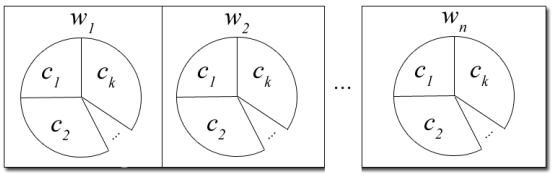

Fig. 6. An encoded particle with n dimensions, each with k classifier ids

Figure 6 shows an encoded particle P. Each section labelled $c_1,c_2..c_k$ (in each dimension $w_1,w_2,..,w_n$) represents an amount of pheromone. The probability of choosing each classifier c_i (i=1,..,k) in each dimension w_d (d=1,..,n) is proportional to the amount of pheromone (τ) in the corresponding pheromone entry τ_{pdi}.

The "decoded" state is then evaluated, and if it is better than the previous personal best state (P_b), it is set as the personal best state for the particle. A particle finds its best neighbour (N) according to the fitness of each neighbour's best state (P_b). In this paper the particles are arranged in a Von-Neumann topology [6], so that each particle has four neighbours.

A slightly different pheromone updating approach is taken with the PSO/ACO algorithm for classifier selection when compared to the PSO/ACO algorithm for rule discovery. As detailed in the pheromone updating procedure in Pseudocode 1, the approach simply consists of adding an amount of pheromone proportional to f to the pheromone entry corresponding to τ_{pdy}. Where f is the fitness of the best neighbour's best state, y is the best neighbour's best state ($c_1,..c_k$) in the particular dimension d ($w_1,..w_n$) and P is the current particle. Although not used in this paper the amount of pheromone added can be modified to slow down (or speed up) convergence, this is achieved using the constant $á$. The closer this constant is set to 0 the slower the convergence achieved. The pheromone vectors are normalised after pheromone has been added, so that the pheromone entries of each pheromone vector add up to 1.

5 Experimental Setup

5.1 Bioinformatics Data Sets

The hierarchical classification methods discussed above were evaluated in four challenging datasets involving the prediction of protein function. The protein functional classes to be predicted in these data sets are the functional classes of GPCRs (G-Protein-Coupled Receptors). GPCRs [9] are proteins involved in signalling. They span cell walls so that they influence the chemistry inside the cell by sensing the chemistry outside the cell. More specifically, when a ligand (a substance that binds to a protein) is received by the part of the GPCR on the outside of the cell, it (usually) causes an attached G-protein to activate and detach. GPCRs are very important for medical applications because 40%-50% of current drugs target GPCR activity [9].

Predicting GPCR function is particularly difficult because the types of function GPCRs facilitate are extremely varied, from detecting light to managing brain chemistry.

The GPCR functional classes are given unique hierarchical indexes by [10]. The GPCRs, examples (proteins) have up to 5 class levels, but only 4 levels are used in the datasets created in this work, as the data in the 5th level is too sparse for training – i.e., in general there are too few examples of each class at the 5th level. In any case, it should be noted that predicting all the first four levels of GPCR's classes is already a challenging task. Indeed, most works on the classification of GPCRs limit the predictions to just one or two of the topmost class [11], [12], [13], [14].

The data sets used in our experiments were constructed from data in UniProt [15] and GPCRDB [10]. UniProt is a well known biological database, containing sequence data and a rich annotation about a large number of proteins. It also has cross-references for other major biological databases. It was extensively used in this work as a source of data for creating our data sets. Only the UniProtKB/Swiss-Prot was used as a data source, as it contains a higher quality, manually annotated set of proteins. Unlike Uniprot, GPCRDB is a database specialised on GPCR proteins.

We performed experiments with four different kinds of predictor attributes, each of them representing a kind of "protein signature", or "motif", namely: FingerPrints from the Prints [17] database, Prosite patterns [16], Pfam [18] and Interpro entries [19]. The four GPCR data sets each use predictor attributes from one of either the Prints, Prosite, Interpro or Pfam databases. They also contain two additional attributes, namely the protein's molecular weight and sequence length.

Any duplicate examples (proteins) in a data set are removed in a pre-processing step, i.e., before the hierarchical classification algorithm is run, to avoid redundancy. If there are fewer than 10 examples in any given class in the class tree that class is merged with its parent class. If the parent class is the root node, the entire small class is removed from the data set. This process ensures there is enough training and test data per class to carry out the experiments. (If a class had less than 10 examples, during the 10-fold cross-validation procedure there would be at least one iteration where there would be no example of that class in the test set).

After data pre-processing, the final datasets used in the experiments have the numbers of attributes, examples (proteins) and classes per level (expressed as level 1/ level 2/level 3/level 4) indicated in Table 1.

Table 1. Main characteristics of the datasets used in the experiments

	GPCR/Prints	GPCR/Prosite	GPCR/Interpro	GPCR/Pfam
#Attributes	283	129	450	77
#Examples	5422	6261	7461	7077
#Classes	8/46/76/49	9/50/79/49	12/54/82/50	12/52/79/49

5.2 Data Set Partitioning and Algorithmic Details

The data sets were split into two main subsets at each iteration of the 10-fold cross validation process, one test set and one training set. The test set is used to assess the performance of the approach in question; therefore the true class of each test example remains unseen

during the training process, only to be revealed to measure the predictive accuracy of the approach. The training set is split into a further two subsets. Firstly 75% of the training set was used as the building set; this building set is used to train the classifiers. Secondly the validation set, which consists of the remaining 25% of the training examples. The validation set is used to compute the quality of the classifiers, and so particle fitness in the PSO/ACO algorithm. After the best solution (according to accuracy in the validation set) has been found in a single PSO/ACO run, the classifiers at every classifier node specified in that best particle are trained using the entire training set. This procedure attempts to maximise the individual classifier's accuracy and so the final accuracy in the test set.

As a baseline it is important to evaluate the proposed method by comparing its predictive accuracy with the predictive accuracy of the greedy selective top-down approach. The baseline should also include each of the individual classification algorithms used in the greedy selective top-down approach. Therefore the first experiments are to build standard TDDC trees using one type of classification algorithm throughout.

The classification algorithms used in the experiments presented in this paper were implementations from the WEKA [20] package. These algorithms were chosen to include a diverse set of paradigms, while having high computational efficiency:

- HyperPipes is a very simple algorithm that constructs a "hyperpipe" for every class in the data set; each hyperpipe contains each attribute-value found in the examples from the class it was built to cover. An example is classified by finding which hyperpipe covers it the best.
- NaiveBayes uses Bayes' theorem to predict which class an example most likely belongs to, it is naïve because it assumes attribute independence.
- J48 is a decision tree algorithm, being WEKA's modified version of the very well known C4.5 algorithm.
- ConjunctiveRule is another very simple algorithm that only produces two rules to classify the entire data set. A "default" rule is produced that predicts the class with the greatest numbers of records in the training set. The other rule is constructed using information gain to select attribute-values for the antecedent.
- BayesNet uses a Bayesian network to classify examples and can theoretically completely take into account attribute dependency.

Although some of these algorithms are clearly more advanced than the others, all were selected for some classifier nodes by the classifier selection method (greedy approach or PSO/ACO) during training, confirming that all of them perform best in certain circumstances. All experiments were performed using 10-fold cross validation [20] with $\acute{a}$ set to 1 for the PSO/ACO algorithm.

6 Computational Results

The predictive accuracy for each method (the five baseline clasifiers used thoughout the TDDC tree, the greedy and PSO/AOCO methods for classifier selection) are shown in Tables 2 through 5 for each dataset. The values after the "±" symbol are standard deviations (calculated using the WEKA statistics classes). Tables 2 through 5 are shown for the sake of completeness, but, to simplify the analysis (and due to paper size restrictions) we focus mainly on a summary of the results (Table 6). Table 6 shows the

summary of the number of cases where there is a statistically significant difference in the predictive accuracy of the 2 methods according to the WEKA corrected two-tailed student t-test (with a significance level 1%). Each cell shows the number of times the labelled approach (Greedy or PSO/ACO) significantly beats the baseline classification algorithm (HP – HyperPipes, NB – NaiveBayes, CR – ConjunctiveRule, BN – BayesNet), in each data across all four class levels. Totals across all data sets are shown at the bottom of the table.

Table 2. Percentage accuracy for each approach in the Prints data set

TDDC Type	Percentage accuracy at each level in the class hierarchy			
	1st	2nd	3rd	4th
HyperPipes	90.76±0.34	76.79±0.55	49.99±1.1	75.42±2.11
NaiveBayes	87.74±0.71	72.72±1.11	41.3±0.99	63.85±1.89
J48	91.68±0.51	83.35±1.0	58.34±1.26	85.14±1.8
ConjunctiveRule	80.16±0.31	49.63±0.46	17.03±0.84	24.8±0.87
BayesNet	88.34±1.39	77.41±1.25	48.0±0.93	74.53±2.94
Greedy	91.68±0.51	83.06±0.88	58.21±1.23	84.66±2.09
PSO/ACO	91.59±0.52	82.67±1.13	57.99±1.52	84.8±2.34

Table 3. Percentage accuracy for each approach in the Interpro data set

TDDC Type	Percentage accuracy at each level in the class hierarchy			
	1st	2nd	3rd	4th
HyperPipes	83.74±1.14	73.77±1.01	48.21±0.95	82.62±2.5
NaiveBayes	87.88±0.59	74.78±0.78	38.59±1.07	51.25±1.85
J48	90.36±0.34	80.68±0.66	51.06±0.93	79.86±2.68
ConjunctiveRule	73.68±0.18	47.73±0.48	17.76±0.47	24.84±0.68
BayesNet	89.18±0.67	78.99±0.83	46.4±0.94	67.3±2.62
Greedy	90.36±0.34	80.41±0.81	54.36±1.33	83.58±2.46
PSO/ACO	90.36±0.34	80.4±0.78	54.43±1.27	84.24±2.27

Table 4. Percentage accuracy for each approach in the Pfam data set

TDDC Type	Percentage accuracy at each level in the class hierarchy			
	1st	2nd	3rd	4th
HyperPipes	92.02±0.44	25.4±0.75	9.8±0.82	4.58±1.22
NaiveBayes	89.59±0.72	59.23±1.41	19.6±1.43	16.27±2.39
J48	92.98±0.48	70.77±1.39	37.03±1.07	48.97±3.98
ConjunctiveRule	75.55±0.13	51.4±0.53	13.49±2.0	6.97±4.63
BayesNet	90.35±1.1	62.7±1.45	23.25±1.46	23.43±2.42
Greedy	92.98±0.48	70.54±1.29	36.97±1.2	48.24±3.55
PSO/ACO	92.98±0.48	70.5±1.35	36.97±1.21	48.5±3.58

Table 5. Percentage accuracy for each approach in the Prosite data set

TDDC Type	Percentage accuracy at each level in the class hierarchy			
	1st	2nd	3rd	4th
HyperPipes	82.14±0.71	46.03±1.28	23.1±1.62	32.16±2.82
NaiveBayes	85.34±1.14	60.63±1.25	24.86±1.3	23.94±2.11
J48	84.71±0.57	61.02±1.12	29.31±1.63	39.58±3.35
ConjunctiveRule	78.68±0.15	41.38±0.25	14.79±0.45	10.0±0.89
BayesNet	85.93±0.88	62.17±1.06	26.68±1.35	31.14±2.47
Greedy	85.93±0.88	62.54±0.91	31.46±1.25	40.73±4.21
PSO/ACO	85.93±0.88	62.8±1.33	32.18±1.48	43.11±3.71

Table 6. Summation of the number of statistically significant resutlts

Dataset	Classif. Selection Approach	Classification Algorithm				
		HP	NB	J48	CR	BN
GPCR/Prints	Greedy Selective	4	4	0	4	4
	PSO/ACO	4	4	0	4	4
GPCR/InterPro	Greedy Selective	3	4	1	4	4
	PSO/ACO	3	4	2	4	4
GPCR/Pfam	Greedy Selective	4	4	0	4	4
	PSO/ACO	4	4	0	4	4
GPCR/Prosite	Greedy Selective	4	2	1	4	2
	PSO/ACO	4	3	3	4	2
Totals	Greedy Selective	15	14	2	16	14
	PSO/ACO	15	15	5	16	14

Both the greedy and PSO/ACO approach for classifier selection were very successful in improving predictive accuracy with respect to four of the base classification algorithms (HP, NB, CR, BN), as shown by the totals in Table 6. These two approaches were less successful in improving accuracy with respect to J48, but even in this case the classifier selection approaches improved upon J48's accuracy several times, whilst never decreasing upon J48's accuracy.

The PSO/ACO classifier selection approach significantly improves upon the performance of the greedy approach in four cases overall. PSO/ACO improves on the performance of J48 in five cases, three more than the greedy approach. These improvements are in the third and fourth level of the Prosite dataset and there is also an improvement in the InterPro dataset at the fourth level. As J48 is the hardest classification algorithm to beat, these results show the most difference. However, the PSO/ACO algorithm also scores better against NaiveBayes when compared to the greedy approach in one case – in the Prosite dataset at the second class level.

The results imply that both the PSO/ACO algorithm and greedy approaches benefit more from more "difficult" data sets. The data set in which the base classification algorithms perform worst is the Prosite data set. This data set also yields the biggest

improvement in accuracies when using the greedy (1 significant win over J48), and more so the PSO/ACO (3 significant wins over J48) approach. Indeed for either of these approaches to increase predictive accuracy above that of a base classifier, the base classifier must make an error that is not made by another base classifier. The more mistakes made by a certain classification algorithm (due to a more difficult data set) the higher the probability of another classification algorithm not making the same set of mistakes. Furthermore, it was observed that overfitting is sometimes a limiting factor with the PSO/ACO approach, since increases in validation set accuracy (over the baseline classification algorithms) did not always result in a similar increase in test set accuracy.

7 Conclusions and Future Research

Our experiments show that both the greedy and PSO/ACO approaches for classifier selection significantly improve predictive accuracy over the use of a single fixed algorithm throughout the classifier tree, in the majority of cases involving our data sets. Overall, the PSO/ACO approach was somewhat more successful (significantly better in four cases) than the greedy approach. We believe that the use of a more advanced approach (as discussed in this paper) is more appropriate in more difficult data sets, where classification algorithms are more likely to make mistakes. Estimating a priori how likely a classification algorithm is to make a mistake is an open problem and this topic is left for future research. In this work the proposed PSO/ACO was compared only with Secker et al's greedy selective approach, so one direction for future research is to compare the PSO/ACO with another population-based meta-heuristics for optimisation, e.g. evolutionary algorithms.

References

1. TrEMBL. Visited (June 2007), http://www.ebi.ac.uk/swissprot/sptr_stats/full/index.html
2. Clare, A., King, R.D.: Machine learning of functional class from phenotype data. Bioinformatics 18(1), 160–166 (2007)
3. Secker, A., Davies, M.N., Freitas, A.A., Timmis, J., Mendao, M., Flower, D.: An Experimental Comparison of Classification Algorithms for the Hierarchical Prediction of Protein Function. Expert Update (British Computer Society – Specialist Group on Artificial Intelligence Magazine) 9(3), 17–22 (2007)
4. Holden, N., Freitas, A.A.: Hierarchical Classification of G-Protein-Coupled Receptors with a PSO/ACO Algorithm. In: Proc. IEEE Swarm Intelligence Symposium (SIS 2006), pp. 77–84. IEEE, Los Alamitos (2006)
5. Holden, N., Freitas, A.A.: A hybrid PSO/ACO algorithm for classification. In: Proc. of the GECCO-2007 Workshop on Particle Swarms: The Second Decade, pp. 2745–2750. ACM Press, New York (2007)
6. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann/ Academic Press (2001)
7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, 2nd edn. (2007)

9. Fillmore, D.: It's a GPCR world. Modern drug discovery 11(7), 24–28 (2004)
10. GPCRDB (2007), http://www.gpcr.org/
11. Bhasin, M., Raghava, G.P.: GPCRpred: An SVM-based method for prediction of families and subfamilies of G-protein coupled receptors. Nucleic Acids Res. 1(32 Web Server issue), 383–389 (2004)
12. Guo, Y.Z., Li, M.L., Wang, K.L., Wen, Z.N., Lu, M.C., Liu, L.X., Jiang, L.: Classifying G protein-coupled receptors and nuclear receptors on the basis of protein power spectrum from fast Fourier transform. Amino Acids 30(4), 397–402 (Epub, 2006)
13. Karchin, R., Karplus, K., Haussler, D.: Classifying G-protein coupled receptors with support vector machines. Bioinformatics 18(1), 147–159 (2002)
14. Papasaikas, P.K., Bagos, P.G., Litou, Z.I., Hamodrakas, S.J.: A novel method for GPCR recognition and family classification from sequence alone using signatures derived from profile hidden Markov models. SAR QSAR Environ Res 14(5-6), 413–420 (2003)
15. UniProt (June 2007), http://www.expasy.UniProt.org/
16. Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., De Castro, E., Langendijk-Genevaux, P.S., Pagni, M., Sigrist, C.J.A.: The PROSITE database. Nucleic Acids Res. 34, D227–D230 (2006)
17. Attwood, T.K.: The PRINTS database: A resource for identification of protein families. Brief Bioinform., 252–263 (2002)
18. Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E.L.L., Studholme, D.J., Yeats, C., Eddy, S.R.: The Pfam protein families database. Nucleic Acids Research 32(Database-Issue), 138–141 (2004)
19. Mulder, N.J., et al.: New developments in the InterPro database. Nucleic Acids Res. 35(Database Issue), D224–D228 (2007)
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)