Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 876746, 12 pages
doi:10.1155/2008/876746

Research Article

Particle Swarm for Attribute Selection in Bayesian Classification: An Application to Protein Function Prediction

Elon S. Correa, Alex A. Freitas, and Colin G. Johnson

Computing Laboratory and Center for Biomedical Informatics, University of Kent, Canterbury CT2 7NF, UK

Correspondence should be addressed to Elon S. Correa, elonsc@yahoo.com

Received 29 July 2007; Revised 26 November 2007; Accepted 10 January 2008

Recommended by Jim Kennedy

The discrete particle swarm optimization (DPSO) algorithm is an optimization technique which belongs to the fertile paradigm of Swarm Intelligence. Designed for the task of attribute selection, the DPSO deals with discrete variables in a straightforward manner. This work empowers the DPSO algorithm by extending it in two ways. First, it enables the DPSO to select attributes for a Bayesian network algorithm; which is more sophisticated than the Naive Bayes classifier previously used by the original DPSO algorithm. Second, it applies the DPSO to a set of challenging protein functional classification data, involving a large number of classes to be predicted. The work then compares the performance of the DPSO algorithm against the performance of a standard Binary PSO algorithm on the task of selecting attributes on those data sets. The criteria used for this comparison are (1) maximizing predictive accuracy, and (2) finding the smallest subset of attributes.

1. INTRODUCTION

Most of the particle swarm algorithms present in the literature deal only with continuous variables [1–3]. This is a significant limitation, because many optimization problems are set in a search space featuring discrete variables. Typical examples include problems which require the ordering or arranging of discrete variables, such as scheduling or routing problems [4]. Therefore, the design of particle swarm algorithms that deal directly with discrete variables is pertinent to this field of study.

The work in [5] proposed a discrete particle swarm optimization (PSO) algorithm for attribute selection in Data Mining. Hereafter, this algorithm will be refereed to as the discrete particle swarm optimization (DPSO) algorithm. The DPSO deals directly with discrete variables, and its population of candidate solutions contains particles of different sizes—the DPSO forces each particle to carry a constant number of attributes across iterations. The DPSO algorithm interprets the concept of velocity, used in traditional PSO, as
1 "probability;" renders velocity as a proportional likelihood; and uses this information to sample new particle positions. The motivation behind the DPSO algorithm is indeed to introduce a probability-like approach to particle swarm.

Although specifically designed for the task of attribute selection, the DPSO is not limited to this kind of application. By performing a few modifications, one can apply this algorithm to many other discrete optimization problems, such as facility location problems [6].

Many data mining applications involve the task of building a model for predictive classification. The goal of such a model is to classify examples—records or data instances—into classes or categories of the same type. Noise or unnecessary attributes may reduce the accuracy and reliability of a classification or prediction model. Unnecessary attributes also increase the costs of building and running a model—particularly on large data sets. Before performing classification, it is therefore important to select an appropriate subset of "good" attributes. Attribute selection tries to simplify a data set by reducing its dimensionality and identifying relevant underlying attributes without sacrificing predictive accuracy. As a result, it reduces redundancy in the information provided by the attributes used for prediction. For a more detailed review of the attribute selection task using genetic algorithms, see [7].

The main difference between the DPSO and other traditional PSO algorithms is that the particles in the DPSO do not represent points inside an n-dimensional Euclidean space (continuous case) or lattice (binary case) as in the standard PSO algorithms [8]. Instead, they represent a combination of selected attributes. In previous work, the DPSO was used to select attributes for a Naive Bayes (NB) classifier. The resulting NB classifier was then used to predict postsynaptic function in proteins.

The study presented here extends previous work reported in [5, 9] in two ways. First, it enables the DPSO to select attributes for a Bayesian network algorithm, which is more sophisticated than the Naive Bayes algorithm previously used. Second, it increases the number of data sets used to evaluate the PSO from 1 to 6. All the 6 functional classification data sets used have a much greater number of classes to be predicted—in contrast with the postsynaptic data set used in [5] which had just two classes to be predicted.

The work is organized as follows. Section 2 briefly addresses Bayesian networks and Naive Bayes classifier. Section 3 shortly discusses PSO algorithms. Section 4 describes the standard binary PSO algorithm and Section 5 the DPSO algorithm. Section 6 describes the G-protein-coupled receptors (GPCRs) and Enzyme data sets used in the computational experiments. Section 7 reports computational experiments—it also includes a discussion of the results obtained. Section 8 presents conclusions and points out future research directions.

2. BAYESIAN NETWORKS AND NAIVE BAYES

The Naive Bayes (NB) classifier uses a probabilistic approach to assign each record of the data set to a possible class. In this work, the NB classifier assigns a protein of a data set of proteins to a possible class. A Naive Bayes classifier assumes that all attributes are conditionally independent of one another given the class [10].

A Bayesian network (BN), by contrast, detects probabilistic relationships among these attributes and uses this information to aid the attribute selection process.

Bayesian networks are graphical representations of a probability distribution over a set of variables of a given problem domain [11, 12]. This graphical representation is a directed acyclic graph in which nodes represent the variables of the problem and arcs represent conditional probabilistic independencies among the nodes. A directed acyclic graph G is an ordered pair $G = (V, E)$, where V is a set whose elements are called vertices or nodes and E is a set whose elements are called directed edges, arcs, or arrows. The graph G contains no directed cycles—for any vertex $v \in V$, there is no directed path that starts and ends on v.

2 An example of a Bayesian network is as follows. (This
is a modified version of the so-called “Asia” problem, [13],
3 given in [2.5.3].) Suppose that a doctor is treating a patient
who has been suffering from shortness of breath—called dyspnoea. The doctor knows that diseases such as tuberculosis, bronchitis, and lung cancer are possible causes for that. The doctor also knows that other relevant information includes whether the patient is a smoker—increasing the chances of lung cancer and bronchitis—and what sort of air pollution the patient has been exposed to. A positive x-ray would indicate either tuberculosis or lung cancer. The set of variables for this problem and their possible values are shown in Table 1.

Table 1: Bayesian network: nodes and values for the lung cancer problem. L = low, H = high, T = true, F = false, Pos = positive, and Neg = negative.

Node name	Values
Pollution	{L, H}
Smoker	{T, F}
Cancer	{T, F}
Dyspnoea	{T, F}
X-ray	{Pos, Neg}

Figure 1 shows a Bayesian network representing this problem. For applications of Bayesian networks on evolutionary algorithms and optimization problems, see [14, 15].

More formally, let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a multivariate random variable whose components X_i are also random variables. A corresponding lower-case letter x_i denotes an assignment of state or value to the random variable X_i. Parents (X_i) represent the set of nodes—variables or attributes in this work—that have a directed edge pointing to X_i. Let us consider a BN containing n nodes, X_1 to X_n, taken in that order. A particular value of $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ in the joint probability distribution is represented by

$$p(\mathbf{X}) = p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n), \tag{1}$$

or more compactly, $p(x_1, x_2, ..., x_n)$. The chain rule of probability theory allows the factorization of joint probabilities, therefore

$$\begin{aligned} p(\mathbf{X}) &= p(x_1) p(x_2 \mid x_1) \cdots p(x_n \mid x_1, \ldots, x_{n-1}) \\ &= \prod_i p(x_i \mid x_1, \ldots, x_{i-1}). \end{aligned} \tag{2}$$

As the structure of a BN implies that the value of a particular node is conditional only on the values of its parent nodes, (2) may be reduced to

$$p(\mathbf{X}) = \prod_i p(X_i \mid \text{Parents}\,(X_i)). \tag{3}$$

Learning the structure of a BN is an NP-hard problem
[16, 17]. Many algorithms that developed to this end use 4
a scoring metric and a search procedure. The scoring metric evaluates the goodness-of-fit of a structure to the data. The search procedure generates alternative structures and selects the best one based on the scoring metric. To reduce the search space of networks, only candidate networks in which each node has at most k-inward arcs (parents) are considered —k is a parameter determined by the user. In the present work, k is set to 20 ($k = 20$) to avoid overly complex models.

A greedy search algorithm is used to generate alternative structures for the BN starting with an empty network, the greedy search algorithm adds into the network the edge that most increases the score of the resulting network. The search stops when no other edge addition improves the score of the network. Algorithm 1 shows the pseudocode of this generic greedy search algorithm.

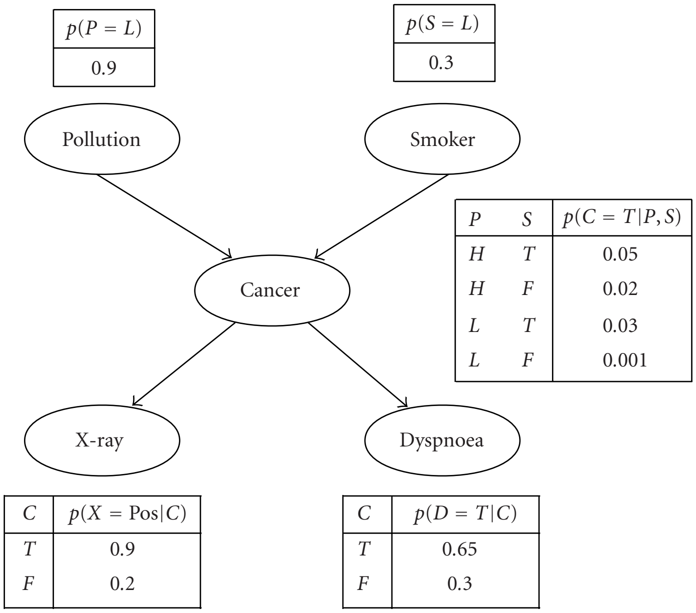

FIGURE 1: A Bayesian network representing the lung cancer problem.

```
Require: Initialize an empty Bayesian network $G$ containing $n$ nodes (i.e., a BN with $n$ nodes but no edges)
Evaluate the score of $G$: $Score(G)$
$BEST = Score(G)$
repeat
    $FROM = 0$
    $TO = 0$
    for $i = 1$ to $n$ do
        for $j = 1$ to $n$ do
            $G' = G$
            if $i \neq j$ then
                if there is no edge between the nodes $i$ and $j$ in $G'$ then
                    Modify $G'$: add an edge between the nodes $i$ and $j$ in $G'$ such that $i$ is a parent of $j$: $(i \rightarrow j)$
                    if the resulting $G'$ is a DAG then
                        if $(Score(G') > BEST)$ then
                            $BEST = Score(G')$
                            $FROM = i$
                            $TO = j$
                        end if
                    end if
                end if
            end if
        end for
    end for
    if $FROM > 0$ then
        Modify $G$: add an edge between the nodes $FROM$ and $TO$ in $G$ such that $FROM$ is a parent of $TO$: (FROM → TO)
    end if
until $FROM = 0$
return $G$ as the structure of the BN
```

ALGORITHM 1: Pseudocode for a generic greedy search algorithm.

To evaluate the "goodness-of-fit" (score) of a network structure to the data, an unconventional scoring metric—specific for the target classification task—is adopted. The entire data set is divided into mutually exclusive training and test sets—the standard methodology for evaluating classifiers, see Section 7.1. The training set is further divided into two mutually exclusive parts. The first part is used to compute the probabilities for the Bayesian network. The second part is used as the validation set. During the search for the best possible network structure, only the validation set is used to compute predictive accuracy. The score of a candidate network is given by the classification accuracy in the validation set. The graphical model of the network that shows the highest predictive accuracy on the validation set—during the entire PSO run—is then used to compute the predictive accuracy on the test set.

Once the best network structure is selected, at the end of the PSO run, the validation set and the other part of the training set are merged and this merged data—that is, the entire original training set—is used to compute the probabilities for the selected Bayesian network. The predicted accuracy—reported as the final result—is then computed on the previously untouched test set. This process is discussed again, in more details, in Section 7.1. A similar process is adopted for the computation of the predictive accuracy using the Naive Bayes classifier.

3. A BRIEF INTRODUCTION TO PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) comprises a set of search techniques, inspired by the behavior of natural swarms, for solving optimization problems [8]. In PSO, a potential solution to a problem is represented by a particle, $\mathbf{Y}(i) = (Y_{(i,1)}, Y_{(i,2)}, \ldots, Y_{(i,n)})$ in an n-dimensional search space. $\mathbf{Y}(i)$ represents the ith particle in the population and n represents the number of variables of the problem. The coordinates $Y_{(i,d)}$ of these particles have a rate of change (velocity) $V_{(i,d)}$, $d = 1, 2, \ldots, n$. Note that the use of the double subscript notation "(i, d)" like in $Y_{(i,d)}$ represents the dth component of the ith particle in the swarm $\mathbf{Y}(i)$—the same rationale is used for $V_{(i,d)}$, and so forth.

Every particle keeps a record of the best position that it has ever visited. Such a record is called the particle's previous best position and denoted by $\mathbf{B}(i)$. The global best position attained by any particle so far is also recorded and stored in a particle denoted by $\mathbf{G}$. An iteration comprises evaluation of each particle, then stochastic adjustment of $V_{(i,d)}$ in the direction of particle $\mathbf{Y}(i)$'s previous best position and the previous best position of any particle in the neighborhood [18]. There is much variety in the neighborhood topology used in PSO, but quite often *gbest* or *lbest* topologies are used. In the *gbest* topology, the neighborhood of a particle consists of all the other particles in the swarm, and therefore all the particles will have the same global best neighbor—which is the best particle in the entire population. In the *lbest* topology, each particle has just a "local" set of neighbors, typically much fewer than the number of particles in the swarm, and so different particles can have different best local neighbors. For a review of the neighborhood topologies used in PSO the reader is referred to [8, 19].

As a whole, the set of rules that govern PSO are evaluate, compare, and imitate. The evaluation phase measures how well each particle (candidate solution) solves the problem at hand. The comparison phase identifies the best particles. The imitation phase produces new particle positions based on some of the best particles previously found. These three phases are repeated until a given stopping criterion is met. The objective is to find the particle that best solves the target problem.

Important concepts in PSO are velocity and neighborhood topology. Each particle, $\mathbf{Y}(i)$, is associated with a velocity vector. This velocity vector is updated at every generation. The updated velocity vector is then used to generate a new particle position $\mathbf{Y}(i)$. The neighborhood topology defines how other particles in the swarm, such as $\mathbf{B}(i)$ and $\mathbf{G}$, interact with $\mathbf{Y}(i)$ to modify its respective velocity vector and, consequently, its position as well.

4. THE STANDARD BINARY PSO ALGORITHM

Potential solutions to the target problem are encoded as fixed size binary strings; that is, $\mathbf{Y}(i) = (Y_{(i,1)}, Y_{(i,2)}, \ldots, Y_{(i,n)})$, where $Y_{(i,j)} \in \{0, 1\}$, $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, n$ [8]. Given a list of attributes $A = (A_1, A_2, \ldots, A_n)$, the first element of $\mathbf{Y}(i)$, from the left to the right hand side, corresponds to the first attribute "A_1," the second to the second attribute "A_2," and so forth. A value of 0 on the site associated to an attribute indicates that the respective attribute is not selected. A value of 1 indicates that it is selected.

4.1. *The initial population for the standard binary PSO algorithm*

For the initial population, N binary strings of size n are randomly generated. Each particle $\mathbf{Y}(i)$ is generated independently. For every position $Y_{(i,d)}$ in $\mathbf{Y}(i)$, a uniform random number φ is drawn on the interval (0, 1). If $\varphi < 0.5$, then $Y_{(i,d)} = 1$, otherwise $Y_{(i,d)} = 0$.

4.2. *Updating the records for the standard binary PSO algorithm*

At the beginning, the previous best position of $\mathbf{Y}(i)$, denoted by $\mathbf{B}(i)$, is empty. Therefore, once the initial particle $\mathbf{Y}(i)$ is generated, $\mathbf{B}(i)$ is set to $\mathbf{B}(i) = \mathbf{Y}(i)$. After that, every time that $\mathbf{Y}(i)$ is updated, $\mathbf{B}(i)$ is also updated if $f(\mathbf{Y}(i))$ is better than $f(\mathbf{B}(i))$. Otherwise, $\mathbf{B}(i)$ remains as it is. Note that $f(\cdot)$ represents the fitness function used to measure the quality of the candidate solutions. A similar process is used to update the global best position $\mathbf{G}$. Once that all the $\mathbf{B}(i)$ have been determined, $\mathbf{G}$ is set to the fittest $\mathbf{B}(i)$ previously computed. After that, $\mathbf{G}$ is updated if the fittest $\mathbf{B}(i)$ in the swarm is better than $\mathbf{G}$. And, in that case, $f(\mathbf{G})$ is set to $f(\mathbf{G}) = f($fittest $\mathbf{B}(i))$. Otherwise, $\mathbf{G}$ remains as it is.

4.3. Updating the velocities for the standard binary PSO algorithm

Every particle $\mathbf{Y}(i)$ is associated to a unique vector of velocities $\mathbf{V}(i) = (V_{(i,1)}, V_{(i,2)}, \ldots, V_{(i,n)})$. Note that, for simplicity, this work uses row vectors rather than column vectors. The elements $V_{(i,d)}$ in $\mathbf{V}(i)$ determine the rate of change of each respective coordinate $Y_{(i,d)}$ in $\mathbf{Y}(i)$, $d = 1, 2, \ldots, n$. Each element $V_{(i,d)} \in \mathbf{V}(i)$ is updated according to the equation

$$V_{(i,d)} = wV_{(i,d)} + \varphi_1 \big(B_{(i,d)} - Y_{(i,d)}\big) + \varphi_2 \big(G_{(d)} - Y_{(i,d)}\big), \quad (4)$$

where w $(0 < w < 1)$, called the inertia weight, is a constant value chosen by the user and $d = 1, 2, \ldots, n$. Equation (4) is a standard equation used in PSO algorithms to update the velocities [20, 21]. The factors φ_1 and φ_2 are uniform random numbers independently generated in the interval (0, 1).

4.4. Sampling new particle positions for the standard binary PSO algorithm

For each particle $\mathbf{Y}(i)$ and each dimension d, the value of the new coordinate $Y_{(i,d)} \in \mathbf{Y}(i)$ can be either 0 or 1. The decision of whether $Y_{(i,d)}$ will be 0 or 1 is based on its respective velocity $V_{(i,d)} \in \mathbf{V}(i)$ and is given by the equation

$$Y_{(i,d)} = \begin{cases} 1, & \text{if } \big(\text{rand} < S(V_{(i,d)})\big), \\ 0, & \text{otherwise;} \end{cases} \quad (5)$$

where $0 \leq \text{rand} \leq 1$ is a uniform random number and

$$S\big(V_{(i,d)}\big) = \frac{1}{1 + \exp\big(-V_{(i,d)}\big)} \quad (6)$$

is the sigmoid function. Equation (5) is a standard equation used to sample new particle positions in the binary PSO algorithm [8]. Note that the lower the value of $V_{(i,d)}$ is, the more likely the value of $Y_{(i,d)}$ will be 0. By contrast, the higher the value of $V_{(i,d)}$ is, the more likely the value of $Y_{(i,d)}$ will be 1. The motivation to use the sigmoid function is to map the interval $[-V_{(i,d)}, V_{(i,d)}]$ for all i, d into the interval (0, 1) which is equivalent to the interval of a probability function.

5. THE DISCRETE PSO (DPSO) ALGORITHM

The DPSO algorithm deals directly with discrete variables (attributes) and, unlike the binary PSO algorithm, its population of candidate solutions contains particles of different sizes. Potential solutions to the optimization problem at hand are represented by a swarm of particles. There are N particles in a swarm. The size of each particle may vary from 1 to n, where n is the number of variables—attributes in this work—of the problem. In this context, the size of a particle refers to the number of different attribute indices that the particle is able to represent at a single time.

For example, given $i, j \in \{1, 2, \ldots, N\}$, in DPSO it may occur that a particle $\mathbf{Z}(i)$ in the population has size 6 ($\mathbf{Z}(i) = \{*, *, *, *, *, *\}$), whereas another particle $\mathbf{Z}(j)$ in the same population has size 2 ($\mathbf{Z}(i) = \{*, *\}$), and so forth, or any other sizes between 1 and n).

Each particle $\mathbf{Z}(i)$ keeps a record of the best position it has ever attained. This information is stored in a separate vector labeled as $\mathbf{B}(i)$. The swarm also keeps a record of the global best position ever attained by any particle in the swarm. This information is also stored in a separate vector labeled $\mathbf{G}$. Note that $\mathbf{G}$ is equal to the best $\mathbf{B}(i)$ present in the swarm.

5.1. Encoding of the particles for the DPSO algorithm

Each attribute is represented by a unique positive integer number, or index. These numbers, indices, vary from 1 to n. A particle is a subset of nonordered indices without repetition, for example, $\mathbf{Z}(k) = \{2, 4, 18, 1\}$, $k \in \{1, 2, \ldots, N\}$.

5.2. The initial population for the DPSO algorithm

The original work on DPSO [5] used a randomly generated initial population for the standard PSO algorithm and a new randomly generated initial population for the DPSO algorithm, when comparing these algorithms' performances in a given data set. However, the way in which those populations were initialized generated a doubt about a possible advantage of one initial population over the other—which would bias the performance of one algorithm over the other. In this work, to eliminate this possible bias, the initial population used by the DPSO is always identical to the initial population used by the binary PSO. They differ only in the way in which solutions are represented. The conversion of every particle in the initial population of solutions of the binary PSO to the Discrete PSO initial population is as follows.

The index of every attribute that has value 1 is copied to the new solution (particle) of the DPSO initial population. For instance, an initial candidate solution for the binary PSO algorithm equal to $\mathbf{Y}(k) = (1, 0, 1, 1, 0)$ is converted into $\mathbf{Z}(k) = \{1, 3, 4\}$ for the DPSO algorithm—because attributes A_1, A_3, and A_4 are set to 1 (are present) in $\mathbf{Y}(k)$, $k \in \{1, 2, \ldots, N\}$. Note that the same initial population of solutions is used to both algorithms, binary PSO and DPSO, to make the comparison between the performances of these algorithms as free from initialization bias as possible.

Initializing the particles $\mathbf{Z}(i)$ in this way causes different particles, in DPSO, to have different sizes. For instance, an initial candidate solution $\mathbf{Y}(j) = (1, 1, 0, 0, 0)$ (from the binary PSO algorithm) is converted into the initial candidate solution $\mathbf{Z}(j) = \{1, 2\}$ (to the DPSO algorithm) which has size 2, whereas another initial candidate solution $\mathbf{Y}(k) = (0, 1, 1, 1, 1)$ (binary PSO) is converted into the initial candidate solution $\mathbf{Z}(k) = \{2, 3, 4, 5\}$ (DPSO) which has size 4, $j, k \in \{1, 2, \ldots, N\}$ and $n = 5$.

In the DPSO algorithm, for simplicity, once the size of a particle is determined at the initialization, the particle will keep that same size during the entire execution of the algorithm. For example, particle $\mathbf{Z}(k) = \{2, 3, 4, 5\}$ above, which has been initialized with 4 indices, will always carry exactly 4 indices, $\mathbf{Z}(k) = \{*, *, *, *\}$. The values of those 4 indices, however, are likely to change every time that the particle is updated.

5.3. Velocities = proportional likelihoods

The DPSO algorithm does not use a vector of velocities as the standard PSO algorithm does. It works with proportional likelihoods instead. Arguably, the notion of proportional likelihood used in the DPSO algorithm and the notion of velocity used in the standard PSO are somewhat similar. DPSO uses $\mathbf{M}(i)$ to represent an array of proportional likelihoods and $M(i,d)$ to represent one of $\mathbf{M}(i)$'s components.

Every particle in DPSO is associated with a 2-by-n array of proportional likelihoods, where 2 is the number of rows in this array and n is the number of columns—note that the number of columns in $\mathbf{M}(i)$ is equal to the number of variables of the problem n.

This is an example of a generic proportional likelihood array

$$\mathbf{M}(i) = \begin{pmatrix} \text{proportional-likelihood-row} \\ \text{attribute-index-row} \end{pmatrix}. \tag{7}$$

Each of the n elements in the first row of $\mathbf{M}(i)$ represents the proportional likelihood that an attribute be selected. The second row of $\mathbf{M}(i)$ shows the indices of the attributes associated with the respective proportional likelihoods.

There is a one-to-one correspondence between the columns of this array and the attributes of the problem domain. At the beginning, all elements in the first row of $\mathbf{M}(i)$ are set to 1, for example,

$$\mathbf{M}(i) = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \tag{8}$$

After the initial population of particles is generated, this array is always updated before a new configuration for the particle associated to it is generated. The updating of the likelihoods $M_{(i,d)}$ is based on $\mathbf{Z}(i)$, $\mathbf{B}(i)$, $\mathbf{G}$ and three constant updating factors, namely, α, β, and γ. The updating factors (α, β, and γ) determine the strength of the contribution of $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$ to the adjustment of every coordinate $M_{(i,d)} \in \mathbf{M}(i)$.

Note that α, β, and γ are parameters chosen by the user. The contribution of these parameters to the updating of $M_{(i,d)}$ is as follows. All indices present in $\mathbf{Z}(i)$ have their correspondent proportional likelihood increased by α. In addition to that, all indices present in $\mathbf{B}(i)$ have their correspondent proportional likelihood increased by β. The same for $\mathbf{G}$ for which the proportional likelihoods are increased by γ.

For instance, given $n = 5$, $\alpha = 0.10$, $\beta = 0.12$, $\gamma = 0.14$, $\mathbf{Z}(i) = \{2,3,4\}$, $\mathbf{B}(i) = \{3,5,2\}$, $\mathbf{G} = \{5,2\}$, and also

$$\mathbf{M}(i) = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \tag{9}$$

the updated $\mathbf{M}(i)$ would be

$$\mathbf{M}(i) = \begin{pmatrix} (1.0) & (1.0+\alpha+\beta+\gamma) & (1.0+\alpha+\beta) & (1.0+\alpha) & (1.0+\beta+\gamma) \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \tag{10}$$

Note that index 1 is absent in $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$. Therefore, the proportional likelihood of attribute 1 in $\mathbf{M}(i)$ remains as it is. In this work, the values used for α, β, and γ were $\alpha = 0.10$, $\beta = 0.12$, and $\gamma = 0.14$. These values were empirically determined in preliminary experiments; but this work makes no claim that these are optimal values. Parameter optimization is a topic for future research. As a whole, these values make the contribution of $\mathbf{B}(i)$ ($\beta = 0.12$) to the updating of the $\mathbf{V}(i)$ a bit stronger than the contribution of $\mathbf{Z}(i)$ ($\alpha = 0.10$); and the contribution of $\mathbf{G}$ ($\gamma = 0.14$) even stronger.

The new updated array $\mathbf{M}(i)$ replaces the old one and will be used to generate a new configuration to the particle associated to it as follows.

5.4. Sampling new particle positions for the DPSO algorithm

The proportional likelihood array $\mathbf{M}(i)$ is then used to sample a new instance of particle $\mathbf{Z}(i)$—the particle associated to $\mathbf{M}(i)$. For this sampling process, a series of operations is performed on the array. To start with, every element of the first row of the array $\mathbf{M}(i)$ is multiplied by a uniform random number between 0 and 1. A new random number is drawn for every single multiplication performed.

To illustrate, suppose that

$$\mathbf{M}(i) = \begin{pmatrix} 1.00 & 1.36 & 1.22 & 1.10 & 1.26 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \tag{11}$$

The multiplied proportional likelihood array would be

$$\mathbf{M}(i) = \begin{pmatrix} (1.00\cdot\varphi_1) & (1.36\cdot\varphi_2) & (1.22\cdot\varphi_3) & (1.10\cdot\varphi_4) & (1.26\cdot\varphi_5) \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \tag{12}$$

where $\varphi_1,\ldots,\varphi_5$ are uniform random numbers independently drawn on the interval (0, 1).

Suppose that this is the resulting array $\mathbf{M}(i)$ after the multiplication

$$\mathbf{M}(i) = \begin{pmatrix} 0.11 & 0.86 & 0.57 & 0.62 & 1.09 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}. \tag{13}$$

A new particle position is then defined by ranking the columns in $\mathbf{M}(i)$ by the values in its first row. That is, the elements in the first row of the array are ranked in a decreasing order of value; and the indices of the attributes—in the second row of $\mathbf{M}(i)$—follow their respective proportional likelihoods. For example, ranking the array $\mathbf{M}(i)$ (shown immediately above) would generate

$$\mathbf{M}(i) = \begin{pmatrix} 1.09 & 0.86 & 0.62 & 0.57 & 0.11 \\ 5 & 2 & 4 & 3 & 1 \end{pmatrix}. \tag{14}$$

The next operation now is to select the indices that will compose the new particle position. After ranking the array $\mathbf{M}(i)$, the first s_i indices (in the second row of $\mathbf{M}(i)$), from left to right, are selected to compose the new particle position. Note that s_i represents the size of the particle $\mathbf{Z}(i)$—the particle associated to the ranked array $\mathbf{M}(i)$.

Suppose that the particle $\mathbf{Z}(i)$ associated to

$$\mathbf{M}(i) = \begin{pmatrix} 1.09 & 0.86 & 0.62 & 0.57 & 0.11 \\ 5 & 2 & 4 & 3 & 1 \end{pmatrix} \tag{15}$$

has size 3 ($\mathbf{Z}(i) = \{*,*,*\}$). That makes $s_i = 3$—note that $\mathbf{Z}(j)$, for instance, may have a different size and consequently a different s_j value. For the $\mathbf{Z}(i)$ above, however, as $s_i = 3$ the first 3 indices from the second row of $\mathbf{M}(i)$ would be selected to compose the new particle position. Based on the array $\mathbf{M}(i)$ given above and $s_i = 3$, the indices (attributes) 5, 2, and 4 would be selected to compose the new particle position, that is, $\mathbf{Z}(i) = \{5, 2, 4\}$. Note that indices that have a higher proportional likelihood are, on average, more likely to be selected.

The updating of $\mathbf{Z}(i)$, $\mathbf{B}(i)$, and $\mathbf{G}$ follows what is described in Section 4.2.

Once the algorithms have been explained, the next section briefly introduces the particular data sets (case studies) used to test the algorithms.

6. CASE STUDY: THE GPCR AND ENZYME DATA SETS USED IN THE COMPUTATIONAL EXPERIMENTS

The experiments involved 6 data sets comprising two kinds of proteins, namely, G-protein-coupled receptors (GPCRs) and Enzymes.

The G-protein-coupled receptors (GPCRs) are a protein superfamily of transmembrane receptors. Their function is to transduce signals that induce a cellular response to the environment. GPCRs are involved in many types of stimulus-response pathways, from intercellular communication to physiological senses. GPCRs are of much interest to the pharmaceutical industry because these proteins are involved in many pathological conditions—it is estimated that GPCRs are the target of 40% to 50% of modern medical drugs [22]

Enzymes are proteins that accelerate chemical reactions—they participate in many processes in a biological cell. Some enzymes are used in the chemical industry and other industrial applications where extremely specific catalysts are required. In Enzyme Nomenclature, enzymes are assigned and identified by an Enzyme Commission (EC) number. For instance, EC 2.3.4 is an enzyme with class value 2 in the first hierarchical class level, class value 3 in the second class level, and so forth. This work uses the GPCRs and EC data sets described in Table 2.

These data sets were derived from the data sets used in [23, 24]. Note that both the GPCR and the Enzyme data sets have hierarchical classes. Each protein in these data sets is assigned one class at the first (top) hierarchical level, corresponding to a broad function, another class at the second level, corresponding to a more specialized function, and another class at the third level, corresponding to an even more specialized function, and so forth. This work copes with these hierarchical classes in a simple way by predicting classes one level at a time, as explained in more detail later.

The data sets used in the experiments involved four kinds of protein signatures (biological "motifs"), namely, PROSITE patterns, PRINTS fingerprints, InterPro entries, and Pfam signatures.

Table 2: GPCR and EC data sets. "Cases" represents the number of proteins in the data set, "Attributes" represents the total number of attributes that describe the proteins in the data set and "L1",. . ., "L4" represent the number of classes at hierarchical class levels 1,. . ., 4 respectively.

			# Classes at			
Data set	Cases	Attributes	L1	L2	L3	L4
GPCR-PRINTS	330	281	8	36	52	44
GPCR-PROSITE	190	127	8	32	32	—
GPCR-InterPro	580	448	12	43	67	46
EC-PRINTS	500	380	6	43	83	—
EC-PROSITE	570	583	6	42	84	—
EC-Pfam	730	706	6	41	92	—

Table 3: Predictive accuracy: binary PSO versus DPSO. Paired two-tailed t-test for the predictive accuracy—significance level 0.05.

Class level	Naive Bayes	Bayesian network
1	$t(9) = 0.467$, $p = 0.651$	$t(9) = 3.407$, $p = 0.007$
2	$t(9) = 2.221$, $p = 0.053$	$t(9) = 3.200$, $p = 0.010$
3	$t(9) = 3.307$, $p = 0.009$	$t(9) = 3.556$, $p = 0.006$

PROSITE is a database of protein families and domains. 6
It is based on the observation that, while there is a huge number of different proteins, most of them can be grouped, on the basis of similarities in their sequences, into a limited number of families (a protein consists of a sequence of amino acids). PROSITE patterns are essentially regular expressions describing small regions of a protein sequence which present a high sequence similarity when compared to other proteins in the same functional family.

In the data sets, the absence of a given PROSITE pattern is indicated by a value of 0 for the attribute corresponding to that PROSITE pattern. The presence of it is indicated by a value of 1 for that same attribute.

PRINTS is a compendium of protein fingerprints. A fingerprint is a group of conserved motifs used to characterize a protein family. In the PRINTS data sets, a fingerprint corresponds to an attribute. The presence of a fingerprint is indicated by a value of 1 for that same attribute; the absence by a 0.

Pfam signatures are produced by hidden Markov models, and InterPro integrates a number of protein signature databases into a single database. In this work, Pfam and InterPro entries also correspond to binary attributes indicating whether or not a protein matches those entries, using the same codification described for the PROSITE patterns and Fingerprints.

The objective of the binary PSO and DPSO algorithms is to classify each protein into its most suitable functional class level. The classification of the proteins is performed in each class level individually. For instance, given protein Υ, at first, a conventional "flat" classification algorithm assigns a class to Υ at the first class level only. Once Υ has been classified at the

Table 4: Number of selected attributes: binary PSO versus DPSO. Paired two-tailed t-test for the number of attributes selected—significance level 0.05.

Class level	Naive Bayes	Bayesian network
1	$t(9) = 7.248$, $p = 4.8\text{E-}5$	$t(9) = 8.2770$, $p = 1.6\text{E-}5$
2	$t(9) = 9.052$, $p = 8.1\text{E-}6$	$t(9) = 14.890$, $p = 1.2\text{E-}7$
3	$t(9) = 6.887$, $p = 7.1\text{E-}5$	$t(9) = 9.1730$, $p = 7.3\text{E-}6$

first class level, the conventional flat classification algorithm is again applied to assign a class to ϒ at the second level—no information about ϒ's class at the previous level is used. The same process is used to assign a class to protein ϒ at the third class level, and so forth.

7. EXPERIMENTS

The quality of a candidate solution (fitness) is evaluated in three different ways: (1) by a baseline algorithm—using all possible attributes; (2) by the binary PSO—using only the attributes selected by this algorithm; and (3) by the discrete PSO (DPSO) algorithm—using only the attributes selected by this algorithm. Each of these algorithms computes the fitness of every given solution using two distinct techniques: (a) using a Naive Bayes classifier; and (b) using a Bayesian network.

7.1. *Experimental methodology*

Note that the computation of the fitness function $f(\cdot)$ for the particles $\mathbf{Y}(i)$ (binary PSO algorithm) and $\mathbf{Z}(i)$ (DPSO algorithm) follows the description given below. For simplicity, only the process using $\mathbf{Y}(i)$ is described—but the same is applicable to $\mathbf{Z}(i)$. $f(\mathbf{Y}(i))$ is equal to the predictive accuracy achieved by the Naive Bayes classifier—and the Bayesian network—on each data set and using only the attributes selected in $\mathbf{Y}(i)$.

The measurement of $f(\mathbf{Y}(i))$ follows a wrapper approach. The wrapper approach searches for an optimal attribute subset tailored to a particular algorithm, such as the Naive Bayes classifier or Bayesian network. For more information on wrapper and other attribute selection approaches, see [25].

The computational experiments involved a 10-fold crossvalidation method [25]. First, the data set being considered is divided into 10 equally sized folds. The folds are randomly generated but under the following criterion. The proportion of classes in every single fold must be similar to the proportion of classes found in the original data set containing all records. This is known as stratified crossvalidation.

Each of the 10 folds is used once as a test set and the remaining of the data is used as training set. Out of the 9 folds in the training set one is reserved to be used as a validation set. The Naive Bayes classifier and the Bayesian network use the remaining 8 folds to compute the probabilities required to classify new examples. Once those probabilities have been computed, the Naive Bayes (NB) classifier and the Bayesian network (BN) classify the examples in the validation set.

The accuracy of this classification on the validation set is the value of the fitness functions $f_{\text{NB}}(\mathbf{Y}(i))$ and $f_{\text{BN}}(\mathbf{Y}(i))$—the same for $f_{\text{NB}}(\mathbf{Z}(i))$ and $f_{\text{BN}}(\mathbf{Z}(i))$. When the run of the PSO algorithm is completed, the 9 folds are merged into a full training set. The Naive Bayes classifier and the Bayesian network are then trained again on this full-training set (9 merged folds), and the probabilities computed in this final, full-training set are used to classify examples in the test set (the 10th fold), which was never accessed during the run of the algorithms.

The reasons for having separate validation and test sets are as follows. In the classification task of data mining, by definition, the goal is to measure predictive accuracy—generalization ability—on a test set unseen during training. Hence, the test set cannot be accessed by the PSO, and is reserved just to compute the predictive accuracy associated with the Bayesian classifier constructed with the best set of attributes selected at the end of the PSO run.

Concerning the validation set, which is used to compute the fitness of particles during the PSO run, this is a part of the original training set which is different from the part of the training set used to build the Bayesian classifier, and the reason for having these two separate parts of the training set is to avoid overfitting of the classifier to the training data; for overfitting in the context of classification, see [7, pages 17, 18]. In other words, if the same training set that was used to build a Bayesian classifier was also used to measure the fitness (accuracy) of the corresponding particle, there would be no pressure to build classifiers with a good generalization ability on data unseen during training, and a classifier could obtain a high accuracy by simply being overfitted to idiosyncrasies of the training set which are unlikely to generalize well to unseen data. By measuring fitness on a validation set separated from the data used to build the classifier, this is avoided, and a pressure to build classifiers with good generalization ability is introduced in the fitness function.

In each of the 10 iterations of the crossvalidation procedure, the predictive accuracy of the classification is assessed by 3 different methods as follows.

(1) *Using all possible original attributes*: all possible attributes are used by the Naive Bayes classifier and the Bayesian network—there is no attribute selection.

(2) *Standard binary PSO algorithm*: only the attributes selected by the best particle found by the binary PSO algorithm are used by the Naive Bayes classifier and the Bayesian network.

(3) *DPSO algorithm*: only the attributes selected by the best particle found by the DPSO algorithm are used by the Naive Bayes classifier and the Bayesian network.

Since the Naive Bayes and Bayesian network classifiers used in this work are deterministic, only one run—for each of these algorithms—is performed for the classification using all possible attributes.

For the binary PSO and the DPSO algorithms, 30 independent runs are performed for each iteration of the crossvalidation procedure. The results reported are averaged over these 30 independent runs and over the 10 iterations of the crossvalidation procedure.

The population size used for both algorithms (binary PSO and DPSO) is 200 and the search stops after 20 000 fitness evaluations—or 100 iterations.

The binary PSO algorithm uses an inertia weight value of 0.8 (i.e., $w = 0.8$). The choice of the value of this parameter was based on the work presented in [26].

Other choices of parameter values for the DPSO were $\alpha = 0.10$, $\beta = 0.12$, and $\gamma = 0.14$, chosen based on empirical experiments but probably not optimal values.

The measurement of the predictive accuracy rate of a model should be a reliable estimate of how well that model classifies the test examples—unseen during the training phase—on the target problem.

In Data Mining, typically, the equation

$$\text{standard accuracy rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{16}$$

is used to assess the accuracy rate of a classifier—where TP, TN, FP, FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively [25].

However, if the class distribution is highly unbalanced, (16) is an ineffective way of measuring the accuracy rate of a model. For instance, in many problems, it is easy to achieve a high value for (16) by simply predicting always the majority class. Therefore, on the experiments reported on this work, a more demanding measurement for the accuracy rate of a classification model is used.

This measurement has been used before in [27]. It is given by the equation

$$\text{predictive accuracy rate} = \text{TPR} \cdot \text{TNR}, \tag{17}$$

where, TPR = TP/(TP+FN) and TNR = TN/(TN+FP)—TPR stands for true positive rate and TNR stands for true negative rate.

Note that if any of the quantities TPR or TNR is zero, the value returned by (17) is also zero.

7.2. *Discussion*

Computational results are reported in Tables 5 and 6. Let us focus the discussion on the results obtained by the 3 algorithms (binary PSO, DPSO, and baseline algorithm) for attribute selection on the GPCR-PROSITE data set, see Table 5. The results obtained for the other 5 data sets are similar. To start with, the results obtained using the Naive Bayes classifier are presented.

Results obtained using the Naive Bayes classifier approach

To assess the performance of the algorithms, two criteria were considered: (1) maximizing predictive accuracy; and (2) finding the smallest subset of attributes.

The results for the first criterion, accuracy, show that both versions of the PSO algorithm did better—in all class levels—than the baseline algorithm using all attributes.

Furthermore, the DPSO algorithm did slightly better than the binary PSO algorithm also in all class levels. Nevertheless, the difference in the predictive accuracy performance between these algorithms is, in some cases, statistically insignificant.

Table 3 shows the results of a paired two-tailed t-test for the predictive accuracy of the binary PSO versus the predictive accuracy of the DPSO at a significance level of 0.05.

Table 3 shows that, using Naive Bayes as classifier, the only statistically significant difference in performance—in terms of predictive accuracy—between the algorithms binary PSO and DPSO is at the third class level. By contrast, using Bayesian networks as classifier, the difference in performance is statistically significant at all class levels.

Nevertheless, the discriminating factor between the performance of these algorithms is on the second comparison criterion—finding the smallest subset of attributes.

The DPSO not only outperformed the binary PSO in predictive accuracy, but also did so using a smaller subset of attributes in all class levels. Moreover, when it comes to effectively pruning the set of attributes, the difference in performance between the binary PSO and the DPSO is always statistically significant, as Table 4 shows.

Results obtained using the the Bayesian network approach

Again, the predictive accuracy attained by both versions of the PSO algorithm surpassed the predictive accuracy obtained by the baseline algorithm in all class levels.

DPSO obtained the best predictive accuracy of all algorithms in all class levels. Regarding the second comparison criterion, finding the smallest subset of attributes, again DPSO always selected the smallest subset of attributes in all hierarchical levels.

The results on the performance of the classifiers—Naive Bayes versus Bayesian networks—show that Bayesian networks did a much better job. For all class levels, the predictive accuracy obtained by the 3 approaches (baseline, binary PSO and DPSO) using Bayesian networks was significantly better than the predictive accuracy obtained using Naive Bayes classifier. The Bayesian networks approach also enabled the two PSO algorithms to do the job using fewer selected attributes—compared to the Naive Bayes approach.

The results emphasize the importance of taking relationships among attributes into account—as Bayesian networks do—when performing attribute selection. If these relationships are ignored, predictive accuracy is adversely affected.

The results also show that for all 6 data sets tested, the DPSO algorithm not only selected the smallest subset of attributes, but also obtained the highest predictive accuracy in every single class level.

8. CONCLUSIONS

Computational results show that the use of unnecessary attributes tends to derail classifiers and hurt classification accuracy. Using only a small subset of selected attributes, the binary PSO and DPSO algorithms obtained better predictive accuracy than the baseline algorithm using all attributes. Previous work had already shown that the DPSO algorithm outperforms the binary PSO in the task of attribute selection [5], but that work involves only one data set. This current work

Table 5: Results for the GPCRs data sets. For the binary PSO and DPSO algorithms, 30 independent runs are performed. The results reported are averaged over these 30 independent runs. The best result on each line for each performance criterion is marked with an asterisk (*).

		GPCR-PRINTS (281 attributes)				
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	72.36 ± 2.64	73.10 ± 2.71	*73.98 ± 3.13	97.40 ± 1.26	*73.30 ± 4.35
	2	35.56 ± 2.56	37.10 ± 3.10	*40.74 ± 5.36	130.30 ± 1.34	*117.30 ± 6.04
	3	27.00 ± 1.82	29.05 ± 2.71	*31.55 ± 4.54	171.10 ± 3.93	*145.70 ± 3.80
	4	24.26 ± 1.75	26.97 ± 2.24	*30.14 ± 3.78	165.00 ± 4.11	*141.30 ± 5.21
Bayesian network	1	88.67 ± 1.72	89.46 ± 1.73	*89.97 ± 2.11	89.30 ± 3.77	*63.80 ± 3.99
	2	53.46 ± 1.40	56.75 ± 2.47	*58.91 ± 3.95	123.70 ± 3.89	*103.00 ± 4.27
	3	38.93 ± 1.92	43.08 ± 3.03	*50.33 ± 6.45	158.20 ± 4.21	*134.50 ± 4.60
	4	28.47 ± 1.90	30.56 ± 2.63	*39.52 ± 5.32	152.60 ± 3.53	*126.80 ± 4.59
		GPCR-PROSITE (127 attributes)				
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Discrete PSO	Binary PSO
Naive Bayes	1	71.27 ± 2.08	72.88 ± 2.40	*73.05 ± 2.31	85.60 ± 2.84	*74.90 ± 3.48
	2	30.00 ± 2.10	31.34 ± 2.47	*32.60 ± 2.31	101.50 ± 3.14	*83.80 ± 4.64
	3	20.47 ± 0.96	21.47 ± 1.16	*23.25 ± 1.08	102.30 ± 3.77	*87.50 ± 4.25
Bayesian network	1	78.05 ± 2.33	79.03 ± 2.57	*80.54 ± 2.46	78.50 ± 3.50	*65.50 ± 3.41
	2	39.08 ± 2.67	40.31 ± 2.85	*43.24 ± 4.67	94.10 ± 3.70	*73.30 ± 2.67
	3	24.70 ± 1.83	26.14 ± 2.11	*28.97 ± 2.77	94.90 ± 3.90	*77.60 ± 4.35
		GPCR-INTERPRO (448 attributes)				
		Average predictive accuracy			Average number of selected attributes	
Method	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	54.17 ± 2.26	55.33 ± 2.36	*56.55 ± 2.61	136.40 ± 1.17	*120.70 ± 5.01
	2	25.19 ± 1.50	26.08 ± 1.62	*27.27 ± 1.87	158.60 ± 1.07	*136.20 ± 5.53
	3	20.03 ± 0.65	21.19 ± 1.03	*22.03 ± 1.63	203.60 ± 1.26	*162.40 ± 6.62
	4	27.97 ± 1.13	29.95 ± 2.18	*30.43 ± 2.56	168.00 ± 0.94	*150.10 ± 7.31
Bayesian network	1	86.68 ± 2.99	89.20 ± 3.06	*89.49 ± 4.28	122.60 ± 4.03	*107.70 ± 5.12
	2	61.85 ± 1.71	64.57 ± 1.43	*68.66 ± 3.82	146.80 ± 3.12	*128.40 ± 5.02
	3	40.77 ± 2.13	44.11 ± 2.48	*46.51 ± 3.41	184.60 ± 2.41	*148.10 ± 3.98
	4	34.05 ± 1.64	36.89 ± 2.56	*39.03 ± 3.63	149.70 ± 2.41	*131.50 ± 4.12

shows much stronger evidence for the effectiveness of DPSO in 6 data sets. In addition, the 6 data sets mined in this work are much more challenging than the two-class data set mined in [5], because the former have several hierarchical class levels per data set, leading to a much larger number of classes to be predicted for each data set.

Even when the difference in predictive accuracy is insignificant, by selecting fewer attributes than the binary PSO, the DPSO certainly enhances computational efficiency of the classifier and is therefore preferable.

The original work on DPSO [5] questioned whether the difference in performance between these two algorithms was attributable to variations in the initial population of solutions. To overcome this possible advantage/disadvantage for one algorithm or the other, the present work used the same initial population for both algorithms.

The results demonstrate that, even using an identical initial population of particles, the DPSO is still outperforming the binary PSO in both predictive accuracy and number of selected attributes. The DPSO is arguably not too different from traditional PSO but still the algorithm has features that enable it to improve over binary PSO on the task of attribute selection.

Another result—although expected—from the experiments is the clear difference in performance between Naive Bayes and Bayesian networks used as classifiers. The Bayesian networks approach outperformed the Naive Bayes approach in all experiments and in all hierarchical class levels.

In this work, the hierarchical classification problem was dealt with in a simple way by "flattening" the hierarchy, that is, by predicting classes for one class level at a time, which permitted the use of flat classification algorithms. The algorithms made no use of the information of the class assigned to a protein in one level to help predict the class at the next hierarchical level. Future work intends to look at an algorithm that makes use of this information.

Table 6: Results for the EC data sets. For the binary PSO and the DPSO algorithms 30 independent runs are performed. The results reported are averaged over these 30 independent runs. The best result on each line for each performance criterion is marked with an asterisk (*).

		EC-PRINTS (380 attributes)				
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributesS	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	72.35 ± 3.33	73.78 ± 3.78	*74.81 ± 3.79	102.80 ± 1.23	*64.20 ± 4.37
	2	31.19 ± 2.26	32.07 ± 2.48	*34.06 ± 2.91	149.00 ± 1.25	*112.30 ± 3.06
	3	23.37 ± 1.73	24.64 ± 2.01	*26.97 ± 2.49	211.10 ± 1.37	*150.60 ± 5.58
Bayesian network	1	88.30 ± 1.94	89.51 ± 2.51	*90.73 ± 2.59	92.80 ± 4.57	*48.90 ± 4.68
	2	53.15 ± 1.49	55.14 ± 1.87	*56.92 ± 2.93	129.70 ± 4.11	*102.00 ± 5.52
	3	36.24 ± 1.62	38.26 ± 2.65	*40.95 ± 4.16	190.40 ± 4.55	*135.10 ± 4.28
		EC-PROSITE (583 attributes)				
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	69.52 ± 5.02	70.37 ± 5.15	*72.31 ± 5.44	118.80 ± 1.14	*98.90 ± 1.85
	2	35.70 ± 1.73	37.73 ± 2.04	*38.83 ± 2.66	154.50 ± 0.85	*134.90 ± 4.93
	3	21.91 ± 1.13	22.86 ± 1.30	*24.36 ± 1.66	197.70 ± 1.16	*154.50 ± 5.34
Bayesian network	1	82.80 ± 1.09	84.83 ± 1.46	*85.95 ± 2.31	105.00 ± 3.62	*92.70 ± 4.00
	2	45.30 ± 2.41	47.82 ± 2.80	*49.50 ± 3.33	135.20 ± 3.65	*119.00 ± 3.89
	3	28.44 ± 2.37	29.40 ± 2.64	*32.52 ± 3.71	172.00 ± 2.94	*146.50 ± 4.40
		EC-PFAM (706 attributes)				
		Average predictive accuracy			Average number of selected attributes	
METHOD	Class level	Using all attributes	Binary PSO	Discrete PSO	Binary PSO	Discrete PSO
Naive Bayes	1	71.61 ± 3.52	72.87 ± 4.02	*74.62 ± 3.77	131.60 ± 5.50	*102.20 ± 3.85
	2	46.70 ± 1.21	48.24 ± 1.39	*49.02 ± 1.17	212.60 ± 5.10	*153.90 ± 5.26
	3	31.00 ± 1.08	32.20 ± 1.53	*33.24 ± 1.76	244.40 ± 4.53	*177.70 ± 2.58
Bayesian network	1	85.94 ± 1.80	87.94 ± 1.80	*89.64 ± 3.27	116.60 ± 4.22	*91.80 ± 4.52
	2	55.34 ± 1.30	56.84 ± 1.49	*58.02 ± 2.02	198.00 ± 4.40	*141.90 ± 4.63
	3	36.56 ± 1.56	37.61 ± 1.44	*39.44 ± 3.07	221.70 ± 4.64	*168.60 ± 4.43

ACKNOWLEDGMENTS

The authors would like to thank Nick Holden for kindly providing them with the biological data sets used in this work. The authors would also like to thank EPSRC (grant *Extended Particle Swarms* GR/T11265/01) for financial support.

REFERENCES

[1] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 489–500, Springer, New York, NY, USA, 2004.

[2] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," in *Proceedings of the 1st European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments (EvoCOP '04)*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 513–524, Springer, Coimbra, Portugal, April 2004.

[3] M. Løvbjerg and T. Krink, "Extending particle swarm optimisers with self-organized criticality," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, et al., Eds., vol. 2, pp. 1588–1593, IEEE Press, Honolulu, Hawaii, USA, May 2002.

[4] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[5] E. S. Correa, A. A. Freitas, and C. G. Johnson, "A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set," in *Proceedings of the 8th Annual Conference Genetic and Evolutionary Computation (GECCO '06)*, M. Keijzer, M. Cattolico, D. Arnold, et al., Eds., pp. 35–42, ACM Press, Seattle, Wash, USA, July 2006.

[6] E. S. Correa, M. T. A. Steiner, A. A. Freitas, and C. Carnieri, "A genetic algorithm for solving a capacity p-median problem," *Numerical Algorithms*, vol. 35, no. 2–4, pp. 373–388, 2004.

[7] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer, Berlin, Germany, 2002.

[8] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[9] E. S. Correa, A. A. Freitas, and C. G. Johnson, "Particle swarm and Bayesian networks applied to attribute selection for protein functional classification," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 2651–2658, London, UK, July 2007.

[10] T. M. Mitchell, *Machine Learning*, McGraw-Hill, London, UK, 1997.

[11] F. V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, New York, NY, USA, 1st edition, 2001.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1st edition, 1988.

[13] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations

with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistics Society*, vol. 50, no. 2, pp. 157–224, 1988.

[14] P. Larrañaga, R. Etxeberria, J. A. Lozano, B. Sierra, I. Naki Inza, and J. M. Peña, "A review of the cooperation between evolutionary computation and probabilistic models," in *Proceedings of the 2nd International Symposium on Artificial Intelligence and Adaptive Systems (CIMAF '99)*, pp. 314–324, La Havana, Cuba, March 1999.

[15] J. M. Peña, J. A. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," *Evolutionary Computation*, vol. 13, no. 1, pp. 43–66, 2005.

[16] R. R. Bouckaert, "Properties of Bayesian belief network learning algorithms," in *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI '94)*, I. R. L. de Mantaras and E. D. Poole, Eds., pp. 102–109, Morgan Kaufmann, Seattle, Wash, USA, July 1994.

[17] D. M. Chickering, D. Geiger, and D. Heckerman, "Learning Bayesian networks is NP-hard," Tech. Rep. MSR-TR-94-17, Microsoft Research, Redmond, Wash, USA, November 1994.

[18] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '97)*, vol. 5, pp. 4104–4109, IEEE, Orlando, Fla, USA, October 1997.

[19] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the Congress of Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3, pp. 1931–1938, IEEE Press, Washington, DC, USA, July 1999.

[20] G. Kendall and Y. Su, "A particle swarm optimisation approach in the construction of optimal risky portfolios," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, part of the 23rd Multi-Conference on Applied Informatics*, pp. 140–145, Innsbruck, Austria, February 2005.

[21] R. Poli, C. D. Chio, and W. B. Langdon, "Exploring extended particle swarms: a genetic programming approach," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 169–176, ACM Press, Washington, DC, USA, June 2005.

[22] D. Filmore, "It's a GPCR world," *Modern Drug Discovery*, vol. 11, no. 7, pp. 24–28, 2004.

[23] N. Holden and A. A. Freitas, "Hierarchical classification of G-protein-coupled receptors with a PSO/ACO algorithm," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '06)*, pp. 77–84, IEEE Press, Indianapolis, Ind, USA, May 2006.

[24] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 100–107, IEEE Press, Pasadena, Calif, USA, June 2005.

[25] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2nd edition, 2005.

[26] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 591–600, Springer, San Diego, Calif, USA, March 1998.

[27] G. L. Pappa, A. J. Baines, and A. A. Freitas, "Predicting postsynaptic activity in proteins with data mining," *Bioinformatics*, vol. 21, supplement 2, pp. ii19–ii25, 2005.

Composition Comments

1. Please check the highlighted changes throughout.

2. We incorporated all footnotes in the text between the highlighted brackets to avoid the use of footnotes as per journal style. Please check.

3. Please rephrase what do you mean by the highlighted part.

4. We added the highlighted word for the sake of clarity and completeness. Please check other highlighted cases throughout.

5. There is an extra right delimeter or missing left one. Please check.

6. We changed "de" to the highlighted "the" in the caption of Table 2 for clarity. Pleas check.