University of Alberta

Biologically inspired modular classifier. ©

by

Michal Kurgan

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of the

Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall, 2008

Your file Votre référence
ISBN: 978-0-494-47284-2
Our file Notre référence
ISBN: 978-0-494-47284-2

Dedication

To my family, especially my parents, for their continuous support, and my brother for his numerous advices and help during my studies. They have put an incredible effort to help me during my education. I especially thank them for continuing words of encouragement and advice.

Abstract

We discuss the use of biologically inspired algorithms, including Genetic Algorithms, Particle Swarm Optimization and Ant Colony Optimization in classification and extraction of classification rules.

Biologically inspired algorithms that are discussed in this work are heuristic-based optimization methods that provide global search strategy and use population of individuals to find approximate solution of a given problem. They provide an interesting alternative for generation of classification rules when compared with traditional greedy search-based approaches. We discuss differences between specific biologically inspired algorithms, including their rule representations, encodings of individuals, their approaches to the rule extraction, and advantages/disadvantages of search strategies that are applied to the classification problems.

We propose enhancements with respect to the rule extraction and rule representation of the current algorithms to introduce new biologically inspired classification algorithm. We examine the properties of generated classification models, which is in contrast to the existing methods that aim at obtaining the highest possible classification accuracy. Instead of focusing on the accuracy, we analyze other properties like the total number of rules, the distribution of rules among specific classes, and modularization of the generated models.

Extensive experimental tests prove that proposed method is comparable or better than compared biologically inspired algorithms in terms of the predictive accuracy, while providing a complete set of modular production rules.

Acknowledgement

I would like to thank all the people who help me to finish this thesis and complete my studies. Especially I appreciate my supervisor Dr. Witold Pedrycz whose experience and support helped me, over the last few years. The understanding and encouragement for my research directions motivated me to continue my research.

I would like to thank the members of my thesis committee, Dr. Marek Reformat, and Dr. Aminah Robinson Fayek for their help during my studies. I also want to thank Dr. Horacio Marquez, the chair of the Electrical and Computer Engineering department.

Table of Contents

List of Tables

List of Figures

1 Introduction

With the current advancements in informational technologies we have witnessed an exponential growth of the amount of stored information. Now it is fairly easy to create customized databases that fit specific user needs and which contain huge amount of easily accessible data. This tremendous amount of information contains potentially useful knowledge, thus the need for data analysts and special (sem-) automatic methods to extract it.

Data mining (DM) is defined as "the nontrivial extraction of implicit, previously unknown and potentially useful information from data" [1]. This is an interdisciplinary field that uses methods from several research areas (including machine learning and statistics) to extract knowledge from the input data. DM is a core step of a broader process called Knowledge Discovery in Databases (KDD), which involves automatic and semiautomatic methods for data analysis and techniques for generation and validation of hidden data structure (hidden knowledge). This process consists of pre-processing methods to facilitate the application of the data mining algorithms, the DM step, and post-processing methods to improve and apply the discovered knowledge [2] [3].

DM algorithms can be divided into two distinctive groups, namely supervised learning algorithms and unsupervised learning algorithms, both referred to as learners. In the case of the latter methods, the data include so called target attribute that defines discrete labels (the corresponding problem is known as classification) or a target attribute that defines continuous values (the problem is known as regression). The main goal is to find relation between the remaining attributes and the values of the target attribute. The unsupervised methods discover the hidden data structure that does not involve any supervision, i.e., a priori knowledge of the labels [4].

The discovered knowledge that encodes the above relation is called a model. The model is often used to predict the values of the target attribute for data that was not used to develop the model. The distinctive difference between different DM algorithms is the structure of the model, it can be used to categorize the learners into groups. The supervised learning algorithms include statistical methods (e.g., Bayesian theory based methods [5], in which case the model is based on a set of probabilities), kernel based algorithms (e.g., Support Vector Machine [6], where the model is based on a set of nonlinear hyperplanes), decision rule, tree, and list induction methods (e.g., RIPPER [7], C4.5 [8], PART [9], respectively, where the model is expressed as a set of rules, some of which can be represented as trees), instance based algorithms (e.g., Nearest Neighbour and k-Nearest Neighbours [5], where the data constitutes the model), and neural networks (e.g., multiple layer perceptron [10] and RBF neural network [11], where the model consists of a network of interconnected processing units called neurons). The unsupervised algorithms include clustering and association rule mining methods.

In this work we focus on the supervised learning algorithms. The supervised learners can be separated in two groups, "white-box" and "black-box", based on the ability of the user to interpret the model. In the former case the model can be read, interpreted, and modified (if desired) by the human user. The "white-box" supervised learners include decision rule, tree, and list induction methods. The latter models are not interpretable by the human user, i.e., only the computer program can handle their complex structure. In our work we focus on the "white-box" learners as we aim to produce human-readable classification models.

Biologically inspired algorithms is a category of algorithms that imitate the nature in the context of the evolution of organisms (e.g. Genetic Algorithms [12], where a group of individuals evolve to create and adapt to an environment), social behaviour of a group of beings, such as fish schools and bird flocks (e.g. Particle Swarm Optimization [13][14], where group of individuals move through an environment and cooperate with each other in order to find locations of food sources), and cooperation between insects such as ant colonies (e.g., Ant Colony Optimization [15], where individuals try to find the shortest path between a colony and a food source). Biologically inspired algorithms introduce a set of behaviour description rules, a set of simple organisms that adhere to the these rules, and a method to iteratively apply the rules. Although the rules are conceptually simple, the algorithms complexity increases with passing iterations. When compared to traditional greedy search-based approaches they provide an interesting alternative which addresses heuristic-based global search strategy.

Recently the biologically inspired algorithms, including the Neural Networks, Genetic Algorithms, Particle Swarm Optimization, and Ant Colony Optimization based algorithms, were applied in the classification domain. In this work we focus on the "white-box" learners. In particular, we consider extending the biologically inspired methods and we contrast our solution with other "white-box" methods. The selected biologically inspired classification methods include Genetic Algorithm based approaches that extract a set of rules [16] and a more recent Particle Swarm Optimization based algorithm, PSO II [17], and an Ant Colony Optimization based algorithm, AntMiner [18]; the latter two methods also generate a set of rules.

In this work we compare the current "white-box" biologically inspired classification algorithms such as PSO II and AntMiner and we propose extensions to the considered algorithms. We discuss the design of the new "white-box" biologically inspired classification method and its advantages and disadvantages in comparison with the current algorithms. We experimentally compare the proposed algorithm with the selected biologically inspired classification algorithms and other "white-box" supervised learners, including decision rule, tree, and list induction methods such as RIPPER, C4.5, PART, respectively.

The proposed biologically inspired algorithm has the following characteristics:

- It improves the quality of the current biologically inspired classification algorithms and provides quality comparable to traditional "white-box" classification algorithms.
- It's classification model consist of a set of modular rules, i.e., the rules are independent and they can be used separately, which is in contrast to the current biologically inspired classification algorithms.
- The classification model provides rules for each class in a given problem, while in case of existing "white-box" methods some classes may not have the

associated rules in the classification model.

Main goals and the anticipated contributions of this work follow:

- Development of novel rule representations in context of biologically inspired classification algorithms.
- Development of novel rule extraction procedure which extends current procedures used in the biologically inspired algorithms.
- Development of novel classification procedure in context of biologically inspired algorithms.
- Design, implementation and comparison (with other "white-box" methods) of the proposed biologically inspired classification algorithm.

This document is organized as follows. Chapter 1 contains necessary background and definitions related to the problems discussed in this work. We discuss not only definitions related to classification, but also provide information about biologically inspired algorithms such as Genetic Algorithms [12], Particle Swarm Optimization [13] [14] and Ant Colony Optimization [15]. Existing biologically inspired classification algorithms are reviewed in the Chapter 3. In Chapter 4 we introduce the design of the proposed biologically inspired classification algorithm. We do not limit ourselves to the discussion of the proposed method, but we also review advantages and disadvantages of our approach in comparison with the existing algorithms described in Chapter 3. Chapter 5 contains experimental comparison between existing "white-box" classification methods, such as RIPPER [7], C4.5 [8], PART [9], biologically inspired algorithms, such as PSO II [17], AntMiner [18], and the proposed biologically inspired algorithm.

2 Background

This chapter provides an overview of the concepts used throughout this document. It provides necessary background and definitions of terms, including classification and biologically inspired algorithms.

2.1 Classification

The essence of classification is to assign a record (also known as item, case or instance) to one of the discrete labels (also known as class labels). Classifier is a classification algorithm that builds models that discriminate between different class labels. The "white-box" model can be used to discover and analyze potentially interesting knowledge that is hidden in the data.

In a typical supervised scenario, the classifier extracts the model from a training set and the model is evaluated on a set of cases to infer the quality of the generated model. Training and evaluation of the classifier on the same set of cases may produce overestimated results with respect to the quality of the model, which may not reflect the quality of the model in context of its future use. Therefore, the testing should be done on cases that were not seen during model generation. Typically, the cases are split into three disjoint sets that does not contain any common case:

- Training set is a set of cases used to extract the classification model.
- Validation set is a set of cases used to evaluate the classification model during the model generation procedure.

- Test set is a set of cases used to evaluate the final generated classification model.

2.1.1 Attribute

Attributes (also known as variables or features) are record descriptors and are typically of one of the two types: nominal (attribute values are members of an unordered set) or numeric (attribute values are numeric and ordered).

Each record is described by the several attributes, each with a specific domain. One of the attributes is defined as a dependant attribute (also known as class). The remaining attributes are called a predictor attributes. If any of the attributes, including the dependant attribute are unknown (the attribute value is unknown or missing) then the case is referred to as an unknown case and the attribute value is specified as "?".

In the scope of this work the case, the case is represented as follows:

{attribute_1 = value_1, attribute_2 = value_2; …; class = value} (1)

2.1.2 Production rule

Production rule is defined as a conditional clause that involves two parts: the antecedent and the consequent. Rule antecedent contains a combination, typically a conjunction, of conditions on a predictor attribute values, while the rule consequent contains a predicted class label. Structure of the production rule follows:

IF *antecedent* **THEN** *consequent* (2)

In this work we use a general form of the production rule shown in Figure 1. The rule antecedent contains a set of conditions connected by the logical conjunction. Each condition, referred to as a term, is a disjunction of logical tests over the single predictor attribute. The logical tests in two terms cannot involve the same predictor attribute, this means that each attribute can only be used in one or no terms. The rule consequent contains the predicted class label. The production rule format assumes that the predictor attributes are nominal.

```
IF
    attribute_1 = value_1_1 OR attribute_1 = value_1_2 OR … OR attribute_1
= value_1_$N_1$
AND
    attribute_2 = value_2_1 OR attribute_2 = value_2_2 OR … OR attribute_2
= value_1_$N_2$
AND
    …
AND
    attribute_$N$ = value_$N$_1 OR attribute_$N$ = value_$N$_2 OR … OR
attribute_$N$ = value_$N$_$N_N$
THEN class = value
```

Figure 1: Production rule representation, where N is the number of attributes to test and N_i denotes the i-th attribute value.

A given production rule can cover a given case, this means that all terms included in the rule pass logical tests on the case (the term that passes the included logical tests on the case is referred to as passed term), otherwise the case is referred as not covered by the rule, e.g., production rule "IF temperature = low OR temperature = high THEN weather = bad" covers the case {temperature = low; weather = ?} and does not cover the case {temperature = moderate; weather = ?}.

Rule pruning is a technique in DM [19], that removes irrelevant terms or parts of the terms that might have been unduly included in the production rule, i.e., we remove the terms present in the rule antecedent to check whether the modified production rule has better quality than the original production rule. Rule pruning potentially improves the rule, helps avoiding overfitting to the training set, and may simplify the rule.

2.1.3 Verification and validation

The purpose of the validation of a given classification model is to statistically evaluate the performance and quality of the model that was obtained by the model extraction procedure. This can be done using several quality measures such as predictive accuracy, sensitivity, and specificity. These statistical quality measures quantify the performance on a binary classification test (positive class is defined as a class label for which evaluation is performed or the class present in the prediction rule consequent that we currently evaluate, while all other class labels are aggregated together into a negative class). To evaluate the model we use a test set. In a typical supervised learning scenario the class labels of test cases are known and they are used to evaluate the quality of the classifications. However in contrast to the training cases, the test cases are not used during model extraction procedure. We have four possible classification outcomes, see Table 1.

	Predicted class positive	**Predicted class negative**
Original (true) class positive	*true positive* (TP)	*false negative* (FN)
Original (true) class negative	*false positive* (FP)	*true negative* (TN)

Table 1: Binary classification test outcomes.

The *true positive* (TP) indicates the number of correct positive classifications, *true negative* (TN) indicates the number of correct negative classifications, *false positive* (FP) indicates the number of incorrect positive classifications, and *false negative* (FN) indicates the number of incorrect negative classifications.

The evaluation criteria that can be used to quantify the quality of the model are defined as follows:

$$Sensitivity = \frac{TP}{TP+FN} * 100\,\% \tag{3}$$

$$Specificity = \frac{TN}{TN+FP} * 100\,\% \tag{4}$$

$$Predictive\ accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100\,\% \tag{5}$$

The *sensitivity,* see Equation 3, measures the ratio between the number of predicted true positive cases and the of the cases with original (true) positive class, referred to as positive cases, i.e., how many of the positive cases are correctly recognized. The sensitivity estimates the quality of the prediction of the positive data. The *specificity,* see Equation 4, measures the ratio between the the number of predicted true negative cases and the cases with original (true) negative class, referred to as a negative cases, i.e., how many of the negative cases are correctly recognized. Specificity shows how well the prediction model, which was designed for the positive class, excludes the negative class. The *predictive accuracy*, see Equation 5, gives the overall evaluation and is defined as ratio between all correct prediction and all predictions. In this work the predictive accuracy is referred to as accuracy.

2.1.3.1 Cross-validation

Cross-validation is a method to estimate the quality of the classification model. In *n*-fold cross-validation, where *n* indicates number of equally-sized subsets we use *n*-1 subsets to train the model and one remaining subset evaluate the generated classification model. This is repeated *n* times, each time different subset is used as the test set. Finally, we report an average classification evaluation criteria over the training/test experiments [4].

This method is used to report the quality of classification model that can be used to compare different classification algorithms.

2.1.3.2 Paired T-test

Paired T-test, referred to as a T-test, is a statistical hypothesis test that compares two groups of paired samples, which in case of this work correspond to the measured quality of the model. It calculates the differences between each set of pairs and analyzes whether this difference is statistically significant based on the assumption that the groups of samples follow a Gaussian distribution.

In this work the T-test is used to compare the quality of two classification models, which are extracted and tested on the same training and test sets, respectively.

2.2 Biologically inspired algorithms

Biologically inspired algorithms imitate the natural systems such as ant colonies or swarms. These methods bring useful contribution to design of adaptive algorithms that could be used in computer science. In this section we discuss several biologically inspired methods. This list is not complete as we want to focus only on methods, including Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization that were recently applied in the classification domain.

2.2.1 Genetic Algorithms

Genetic Algorithm (GA) [12] is a search technique that is used to find approximate solutions to various optimization problems. It is a population based algorithm that is inspired by the theory of evolution, where number of representatives is maintained and evolved according to principles of natural selection – survival of the fittest individual. This simple concept makes GA easy to implement and adopt to solve problems in different domains [12][16][20][21].

GA is an optimization algorithm that is inspired by the evolutionary biology techniques (which are based on genetic operations such as inheritance, mutation, selection and crossover), where a group of individuals represented by chromosomes evolve toward better solutions. Specific individuals are first selected for mating, and then modified, using crossover and mutation operations, to form new population. This scheme, which is based on the concept of organisms' evolution, aims at improving of the population with time and passing generations.

A typical GA computer simulation requires definition of genetic representation of the solution domain and the fitness function that is used to evaluate candidate solutions. In this approach, each individual chromosome is equivalent to single candidate solution.

Genetic representation of the solution domain does not concern only chromosome interpretation and encoding scheme, but also the genetic operations. Traditionally, a standard representation of a chromosome is a bit string, but other types and structures can be used in a similar way. Regardless of the actual type, a mapping between specific representation and solution domain has to be defined; it provides interpretation of the individual. Additionally, crossover and mutation operations have to be designed for the chosen chromosome type. These two operations are crucial to the GA algorithm as population evolution depends on them. Crossover is an analogous operation to the reproduction; it uses two input chromosomes (current generation) to generate two new chromosomes (next generation). As the outcome, this technique ensures differences between two consecutive generations of individuals. In contrast, mutation process concerns only a single chromosome, usually introducing random changes in its representation. It is analogous to the biological mutation and ensures genetic diversity in the population.

Fitness function is used to evaluate chromosome and quantify its quality in a given problem domain. This value is used as a measure to select the best representatives of the current generation population that are used to create the next generation. The quality measure is a driving force for population evolution towards better, possibly optimal problem solutions.

In the GA algorithm, population evolves from the initial set of chromosomes. Using selection, crossover and mutation operations, population should improve its quality, as measured with the fitness function, with passing time and generations. GA is an iterative algorithm, where single iteration corresponds to a certain generation. In each iteration, see Figure 2, first a number of population representatives (chromosomes) is selected. The selection is based on the quality of the members, only the fittest ones are used to obtain the next generations. Additionally, in order to avoid destroying good solutions/individuals, a technique called elitism may be used. In elitism, a pre-defined number of best population members is automatically passed to the next generation; this means that these selected individuals are not subjected to crossover and mutation operations. When a set of best population representatives is specified, we use the crossover and mutation operations to create the next generation individuals. The last step of the algorithm is the stop condition. We usually check whether the maximum number of generations was exceeded, but the stop condition can be defined as a convergence criterion, which is based on the evaluation of the current generation or a few of the last generations.

```
initialize_population()
LOOP
    set = select_chromosomes(population)
    crossover_chromosomes(set)
    mutate_chromosomes(set)
    update_population(population, set)
UNTIL stop_condition()
```

Figure 2: Pseudo-code of the GA algorithm

2.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [13][14][22] is a Swarm Intelligence [22] based global optimization method. Initially, it was introduced as an algorithm to solve continuous problems, but extensions and modifications make it possible to apply this method to discrete problems i.e. Binary Particle Swarm Optimization (BPSO) [23]. The PSO algorithm is based on a simple concept and because of that, it can be easily implemented. It uses only primitive mathematical operators that are computationally inexpensive, both in terms of small memory requirements and speed [13].

PSO is an optimization algorithm inspired by the intelligent group behaviour of beings such as birds or fishes and their social experience in community (swarm). Although each individual, often called particle, has limited capabilities, behaviour of the entire swarm is complex and exhibits intelligence. Particle decisions are based on the environment, its own experience (memory) and information from neighbours (social experience). This means that each individual has a limited experience and memorizing capabilities.

An important aspect of the PSO algorithm is the concept of a particle neighbourhood, as it has great impact on knowledge sharing between the particles. Although there are different models, usually the neighbourhood is based on a relative position in the environment (i.e., Euclidean distance between two particles) or social relations expressed as a predefined relationship between particles (i.e., index in the storing array [17]). While choice of the specific neighbourhood model plays important role, we also have to consider its size. The meaningful sizes vary from single nearest neighbour to all particles in the swarm. In the latter case, a particle shares its experience with the entire swarm. This means that each particle knows best position that was visited by all other swarm members. Usually the number of considered neighbours is either two or all particles [17].

In the PSO algorithm particles explore an environment in order to find the best possible position. In each iteration, all particles in the swarm have to decide where to go; the decision is based on quality of the environment (discussed later), and personal and neighbourhood experience. Using this information, particles move from the current to the next position in the search space. The search procedure continues as long as the stop condition is not fulfilled. To implement the stop condition we usually check whether the maximum number of iterations was exceeded or analyze the particles in the swarm, i.e., the stop condition is based on the convergence of the particles. Various PSO modifications may contain some differences, but this main concept is shared between all variants, see Figure 3.

```
initialize_swarm()
LOOP
    FOR particle IN 1:number_of_particles
        evaluate_particle_environment(particle)
        evaluate_particle_experience(particle)
        FOR k IN neighbourhood
            evaluate_particle_neighbour(particle, k)
        END
        move_particle(particle)
        update_experience(particle)
    END
UNTIL stop_condition()
```

Figure 3: Pseudo-code of the PSO algorithm

Each particle position is represented by an array with specified length D that usually equals to the problem dimensionality; it can be measured as the number of attributes or size of the problem solution. In the basic PSO approach single array variable is a floating point value that represents particle position in x_i dimension, but this is not always the case since some PSO algorithm variants may use different data types; e.g. in the BPSO algorithm each particle is a binary string (single dimension is a boolean value).

$$x=(x_{1,}x_{2,}\ldots,x_D), i\in\{1,2,\ldots D\} \tag{6}$$

To simulate particles movement and search capabilities we have to maintain memory for each individual, including current position (x_i), velocity (v_i) and best position (P_i). Additionally depending on the neighbourhood model we have to store best neighbourhood position (P_{ig}) for each particle. Particle movement equations for the PSO algorithm are defined as follows:

$$\begin{aligned} v_i(t) = & \ Xv_i(t-1) \\ & +\varphi_1(P_i - x_i(t-1)) \\ & +\varphi_2(P_{ig} - x_i(t-1)) \\ x_i(t) = & \ x_i(t-1) + v_i(t) \end{aligned} \tag{7}$$

where t is an iteration number, $v_i(t)$ is a particle velocity in *i*-th dimension, $x_i(t)$ is a particle position in *i*-th dimension, X, φ_1, φ_2 are parameters, and P_i, P_{ig} are particle best local and best neighbourhood position in *i*-th dimension.

Particle movement equations for the BPSO algorithm are defined as follows:

$$\begin{aligned} v_i(t) = & \ Xv_i(t-1) \\ & +\varphi_1(P_i - x_i(t-1)) \\ & +\varphi_2(P_{ig} - x_i(t-1)) \\ x_i(t) = & \begin{cases} 1 & rand() < S(v_i(t)) \\ 0 & otherwise \end{cases} \end{aligned} \tag{8}$$

where S is a sigmoid function.

Velocity update equations, $v_i(t)$, are identical for the BPSO and PSO algorithms. In both cases, formula can be separated into three different components. The first part, $Xv_i(t-1)$, represents velocity "memory"; parameter X can be interpreted as an environment resistance during particle movement through search space. The second part, $\varphi_1(P_i - x_i(t-1))$, is a particle "cognition" part, which represents particle private thinking. The third part, $\varphi_2(P_{ig} - x_i(t-1))$, corresponds to the "social" thinking of the particle that describes cooperation between different individuals in the swarm. The two latter parts use φ_1 and φ_2 parameters that are random weights with predefined upper limit; they modify the behaviour of the particle. The parameters allow to choose whether the private thinking is more important or the social behaviour and information sharing with other individuals in the swarm should make precedence.

Because of the different character of the particle position for the standard PSO and the BPSO algorithms, position update equations, $x_i(t)$, are different. For the continuous PSO algorithm, position update is just a simple summation that simulates real life movement. However discrete version of the algorithm uses velocity differently; it is an indication of desired position value. First the velocity is limited to range [0.0; 1.0] using the sigmoid function S, then output value is used as a probability that certain particle position bit equals 1.

During the search procedure, particles move according to equations $x_i(t)$ and $v_i(t)$, using information about the environment. In order to evaluate current position and calculate its quality, a mapping between particle position and problem dependent description is used. This mapping connects the problem that we want to solve with the PSO algorithm. This means that the particle position has to be encoded into specific domain dependent solution.

2.2.3 Ant Colony Optimization

Ant Colony Optimization (ACO) [15][24][25] is a versatile, population based optimization algorithm that can be applied to solve many problems in various domains [26]. ACO was inspired by the collective behaviour of ants and it tries to mimic their group behaviour. It is very interesting that in spite of the almost blindness of ants, they still can find shortest paths between the colony and food sources. Ants communicate with each other using a substance called pheromone that attracts other individuals. During movement, ants drop pheromone on the ground to mark the trail and make it more attractive for other ants. Paths that are used more frequently contain larger amounts of pheromone. As a result, more ants tend to use these selected paths and leave even more pheromone. Using this positive feedback, ants can find shortest paths between points of interest (e.g., food source) and the colony.

Essentially, the algorithm that mimics behaviour of the real world ants is an agent based system that includes methods to cooperate and adapt. In this approach, each problem solution is represented by the path that is constructed by the single agent.

Cooperation simulates group behaviour and information sharing between ants in terms of the use of pheromone trails. Single ant movement is random, except when it encounters already marked path, in which case the path could be detected and followed. Each ant that follows the specific path marks it with additional pheromone, which makes the path more attractive for other ants.

When no pheromone information is available then ants choose the path blindly. In contrast to this behaviour, ACO adapts to the environment and uses quality measure to increase probability to choose better paths. Environment may also be dynamic over time. In this case the algorithm will adapt automatically to the modified environment.

The above mentioned properties of the ACO algorithm imply the following:

- Representation; We represent the problem in such way that its solution can be constructed incrementally. Additionally we have to maintain its validity during construction procedure.

- Cooperation. We provide a pheromone updating scheme that specifies how to modify pheromone trails τ ; both reinforcement (increase) and evaporation (decrease) of the pheromone with time have to be taken into consideration.
- Adaptation. In order to evaluate the solutions quality during construction phase, we provide a problem dependent heuristic function η . This function binds together ACO algorithm with a specific problem that we want to solve and it provides local quality measure of incomplete solutions.
- Movement. During construction phase, ants incrementally constructs given problem solution. In every iteration, each ant decides where to move; basically, this is decision how to extend current candidate solution. It is based on path quality calculated as a combination of the heuristic function (η) and the pheromone trails (τ).

2.2.3.1 Ant System

Ant System was introduced as a first method based on the described concept of Ant Colony Optimization to solve TSP problem [15][24]. In this system, simulated ants behave similarly to their real world counterparts and try to find shortest path between source and destination points. However, TSP problem includes additional constraints: each point can only be visited once and source and destination points are the same. These restrictions have to be taken into consideration during optimization phase to specify available trails for ants, during solution construction. Single solution, an ant path, represents tradesman route between points of interest and have to meet the above requirements.

Ant System is a straightforward algorithm that is based on a simple concept. Main algorithm and solution search scheme is presented in Figure 4. In a single iteration, ants try to construct candidate solutions by moving from the source to the destination points. At this point, pheromone is used to guide search to select possibly better paths. Amount of pheromone that is spread along the paths is updated after an ant finishes its movement, as it was proposed in [24]. However, this step is usually omitted and there is only single pheromone update after all ants construct complete candidate solutions. The final stage of ant processing is to store candidate solution; it may be additionally processed to improve it. The last step is the stop condition; it can be defined as the maximum number of possible iterations or it could be based on the convergence analysis based on the stored candidate solutions [15][24][25].

```
initialize_colony()
LOOP
    FOR ant IN 1:number_of_ants
        initialize_ant(ant)
        path = incrementally_construct_path(ant)
        update_pheromone_local(path)
        update_solutions(path)
    END
    update_pheromone_global(paths)
UNTIL stop_condition()
```

Figure 4: Pseudo code of the ACO algorithm

The most important part of the single Ant System iteration is solution construction

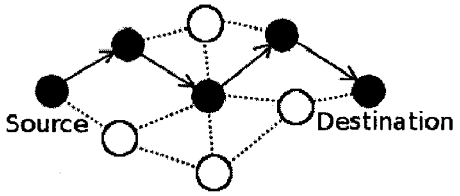


Figure 5: Complete candidate solution constructed by the ant

procedure. It is repeated multiple times, once for each ant in the colony. Initially, each ant starts at the source point and represents an empty solution; next, iteratively, solution is expanded as ant wanders from point to point in search space. Candidate solution is complete when ant arrives to the destination point, at this point construction procedure stops (see Figure 5).

The decision where to move is based both on the pheromone trails τ and the heuristic function η that describes local environment quality. Probability of transition between two points is defined as follows:

$$p_{mn}(t)=\frac{[\eta_{mn}]^{\alpha}[\tau_{mn}(t)]^{\beta}}{\sum_{n\in N}[\eta_{mn}]^{\alpha}[\tau_{mn}(t)]^{\beta}},\ n\in N \tag{9}$$

where t is iteration number, m, n are points in search space, α, β are parameters, and N is an allowed set of points that are connected with point m.

Transition probability is a trade-off between ant visibility (local quality) and pheromone intensity on the trail (the more pheromone is on the path, the more ants are using it) that is controlled by the parameters α and β. Here we implicitly declare that the movement is only possible over trails that are allowed for a specific point. This is very important as we want to construct only valid solutions. In the case of TSP problem, it means that each ant can visit single point only once and we have to dynamically modify set of allowed points during the incremental construction of the solution.

Local quality, described by the heuristic function that tries to estimate the cost of extending solution, is problem dependent. Its main purpose is to guide ants to visit points that are more likely to be included in the optimal problem solution. Heuristic function mainly depends on the problem that we want to solve, e.g. for TSP, a good choice is the function based on the length between nodes.

Pheromone is a communication method that is used for information sharing between ants. As single ant moves and constructs solution, it drops pheromone marks on the visited points. Paths that are used more frequently have more pheromone, thus it is more likely that even more ants would use them. Update of pheromone intensity over paths can be done both locally and globally. The main difference between global and local update is that the former method uses only single path as a reference, while the latter includes all paths that were followed by ants in single Ant System iteration. In both cases, the update formula is similar.

$$\tau_{mn}(t+1)=\rho\tau_{mn}(t)+\Delta\tau_{mn} \tag{10}$$

where t is iteration number, m, n are points in search space, ρ is a evaporation parameter, and $\Delta\tau$ is a reinforcement value

Pheromone intensity update formula takes two aspects into consideration: evaporation and reinforcement. Evaporation, controlled by the parameter ρ, is used to avoid accumulation of pheromone and it simulates behaviour of the substance in real world. Additionally all paths that were visited by all considered ants are reinforced. The increase amount depends both on the number of ants and constructed solutions quality. However, in contrast to heuristic function that describes local environment during construction phase, at this point we evaluate complete solutions to allocate more pheromone for the shorter/better paths.

Although basic ACO algorithm performed well, in case of problems with higher number of dimensions, the exploration of the search space was limited and suffered an early search stagnation (the situation where all ants take same path and thus generate the same solution). To overcome these problems an elitist strategies were proposed [27]. MAX-MIN Ant System introduced few changes to the pheromone handling, including changes to pheromone update formula.

- Pheromone levels should lie within given range $[\tau_{min};\tau_{max}]$. This modification tries to prevent search stagnation in cases when one trail contains significantly more pheromone than all other trails.
- Initialization of pheromone trails with τ_{max} improves exploration of the search space at the beginning of the algorithm.
- Elitist strategy; after each iteration only best solutions are exploited. This means that during pheromone update procedure we do not consider all trails that were created by ants. Instead only selected (usually solutions with highest quality) ones are used.
- Stop condition that uses convergence analysis can be based on pheromone

levels; ACO algorithm stops when pheromone level τ_{max} is present on a single trail and τ_{min} is on all other trails.

2.3 Summary

In this chapter we defined several terms used in the classification domain, including the data, (e.g., the training set, data cases, and attributes), knowledge representation (e.g., the production rules), and methods to quantify extracted knowledge (e.g., the quality measures such as the sensitivity and specificity) and compare classifiers (e.g., the cross-validation and T-test). Additionally, we described biologically inspired algorithms such as Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization and provided details about the heuristic strategies used during the optimization procedure. The concepts discussed in this chapter are used throughout this document.

3 Existing biologically inspired classification algorithms

In this chapter we discuss current biologically inspired classification algorithms. We focus on the recently proposed methods for generation of production rules such as PSO II [17] and AntMiner [18] that are based on Particle Swarm Optimization and Ant Colony Optimization, respectively. While other approaches exist, for instance based on the Genetic Algorithms [16], the above two methods are considered as the representatives of the currently available biologically inspired classification algorithms in terms of the classification algorithm design [16][17][18].

3.1 PSO II

PSO II [17] is the first application of the PSO algorithm in classification domain and it uses the standard PSO algorithm to discover a set of production rules that form the classification model. In this section we discuss the parametrization and the design of the biologically inspired classification algorithm. The next few subsections contain description of the aspects related to the classification model such as rule representation, rule extraction procedure, and classification procedure.

The PSO II algorithm defines the following aspects of the underlying PSO algorithm:

- Particle quality formula that is used to evaluate the current particle position

$$Q(x)=\begin{cases} sensitivity*specificity & if\ 0.0\leq x_i\leq 1.0, i\in 1,2,\ldots,D \\ -1 & otherwise \end{cases} \quad (11)$$

where x_i is *i*-th value of the particle position array, D is the particle position array size, and the sensitivity and specificity are computed for a given particle that represents single production rule over a current training set, i.e., the training cases that are used to extract the production rule with the PSO optimization algorithm.

The particle quality formula penalizes the particles that move out of the range $[0.0; 1.0]$ in any of the particle position array indices.

- Particle movement equation parameters, see Equation 7. The values of X, φ_1, and φ_2 parameters are set to guarantee the convergence of the particle swarm [28][29]. We use the following values: $X=0.73$, $\varphi_1=\varphi_2=1.49$ [17][28][29].
- The PSO algorithm stop condition. The search procedure used to extract single production rule stops when the swarm converges, i.e., the sum of normalized Euclidean distance between all the particles in the swarm is smaller than a user-defined threshold. The value that provides the best results is 0.1 [17].
- The number of particles in the swarm. The size of particle swarm determines a trade-off between computational costs and search capabilities. The bigger the swarm, the more time is needed to run the PSO algorithm, but on the other hand, the number of evaluated points is bigger, thus possibly leading to a better final solution. The number of particles equals to 25 [17].

In the following subsections we discuss several aspects of the PSO II classification algorithm that introduce additional parameters together with their values:

- The rule extraction procedure stop condition. The rule extraction procedure stops when the current training set contains fewer cases than an user-defined parameter The value is set to 5% of cases in the training set.
- The indifference threshold. The parameter is discussed in following subsection. The value equals 0.9 [17].

3.1.1 Rule representation and rule encoding

PSO II algorithm extracts a set of production rules. Single production rule contains only one logical test for each attribute. This means that rule antecedent is a conjunction of logical tests, see Figure 6. In the PSO II algorithm, each term is a single logical test that involves one attribute. This is a simplification of the rule representation shown in Figure 1.

```
IF
    attribute_1 = value_1
AND
    attribute_2 = value_2
AND
    ...
AND
    attribute_N = value_N
THEN class = value
```

Figure 6: Simplified production rule representation in PSO II algorithm, where N is the number of attributes.

An encoding scheme that translates the PSO particle (each particle is an array of floating point numbers of size D) into the relevant production rule is necessary to associate the problem of extracting production rules with the PSO optimization algorithm. In the PSO II rule extraction algorithm D is the number of attributes (including predictor and class attributes), where each particle position array index corresponds to the single term, e.g., if the problem contains 10 attributes (including the class attribute) then the size of the particle position array is 10. The mapping between the array index and the term is defined as follows, see Figure 7:

- Particle search space is limited to range $[0.0; 1.0]$. In case when the particle position violates this constraint for any of the array indices then the particle position is marked as invalid.
- Search space is subdivided into equally sized sub-ranges that map floating point numbers into a specific nominal attribute value, e.g., $[0.0; 0.2)$ maps to the first attribute value, $[0.2; 0.4)$ maps to the second attribute value, etc. This encoding scheme implies that the attribute values are ordered.
- We introduce indifference threshold, which is a user-defined parameter that further divides the search space. It describes probability that a specific attribute is irrelevant and the corresponding term should not be included in the production rule, i.e., in case when the indifference threshold range is $[0.0; 0.5)$ then the rule antecedent does not contain the term testing corresponding attribute if the particle position is within the range. The indifference threshold is used only for these particle position array indexes that map to the predictor

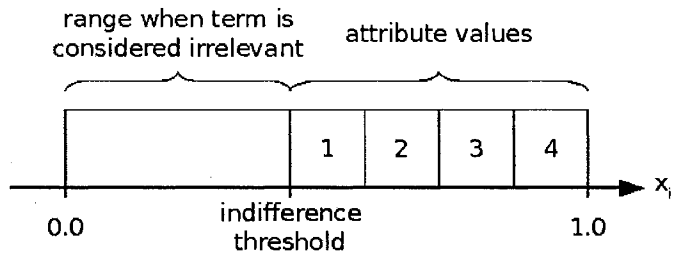


Figure 7: The mapping between the i-th particle position value (x_i) and the term for the predictor attribute with four unique values in the PSO II biologically inspired classification algorithm.

attributes.

The authors claim that the encoding scheme supports the numeric attributes, but the algorithm performance was never tested on the datasets that contain the numeric attributes [17]. In this work we limit ourselves to the analysis of the nominal attributes only. As such we do not define the mapping between the PSO particle and the production rule that contains numeric attributes for the PSO II algorithm.

3.1.2 Rule extraction procedure

Rule extraction procedure is a straightforward training algorithm in which we iteratively create a set of production rules for a given training dataset. Generated ruleset can be used to assign class label to previously unknown cases. All rules are created one after another, in an iterative process, and there is no predefined order that defines how to assign a given class label to the next production rule. Class attribute is considered as a predictor attribute during the optimization procedure and is optimized.

The main algorithm, see Figure 8, starts by removing all unknown data cases, both

```
initialize(pso, data, ruleset)
WHILE number_of_training_cases > max_number_of_uncovered_case
    rule = generate_rule_with_PSO(pso, data)
    update_ruleset(ruleset, rule)
    update_training_dataset(data, rule)
END
post_process_ruleset(ruleset, data)
```

Figure 8: Pseudo-code of the PSO II rule extraction algorithm

from the training and the test sets, and the training set is initialized as the current training set. After this pre-processing step, the production rules are iteratively generated and added to the ruleset. Each iteration consists of a few actions. Namely, the PSO algorithm is a core step and is used to extract a single, best production rule, and next after pruning, the rule is added to the ruleset. Finally, all cases correctly classified by the extracted rule are removed from the current training dataset and next iteration starts. Rule extraction algorithm stops when the remaining number of training cases is lower than a user-defined threshold. The last step of the algorithm is to finalize ruleset, by adding default production rule. The default rule is in the form "**IF** *TRUE* **THEN** class=*default_value*", where *default_value* is the most common class label in the remaining training cases.

The finalized ruleset is subject of additional post processing routine. It involves ruleset cleaning, where all production rules that may never be applied are removed from the ruleset. This procedure includes two types of tests. First, the rules are processed sequentially and we check whether the current rule is superset of the next rule (the next rules contains a subset of the terms of the current rule in the rule antecedent). Second, we check whether the rule predicts the same class label as the default rule (they have the same rule consequent) and is located just before the default rule. The rules that obey one or both of these above tests are removed.

3.1.3 Classification procedure

Ruleset extracted by the PSO II algorithm can be used to classify test cases, which were unseen during training, but in order to do so we have to follow specific procedure. All rules are created, one after another, without any predefined order. As a result of this design, the sequence of production rules and corresponding class labels is arbitrary, and the rules are used in order in which they were generated. The first rule that covers the test case is used and the case is assigned the class predicted by the selected rule consequent.

The ruleset contains a default rule, which is used for all test cases that are not covered by any of the production rules generated during the rule extraction procedure by the PSO optimization algorithm.

3.2 AntMiner

AntMiner [18] is the first method that proposed to use Ant Colony Optimization in the classification domain. In this section we discuss the design of the production rule extraction procedure and discuss the customizations of the ACO algorithm. The next few subsections contain description of several aspects related to the classification model such as rule representation, rule extraction procedure, and classification procedure.

The AntMiner algorithm defines the following aspects of the underlying ACO algorithm:

- Path construction procedure and the stop condition. During the optimization, each ant tries to find shortest path between the source and the destination points. Each point in the constructed path, with exception of the source and the destination points, represents one production rule term. The process of building the production rule is iterative and in each iteration a single term is concatenated to the candidate solution. The ant decision to choose the trail and at the same time the corresponding term is based on the quality of the available trails computed as a combination of the heuristic function (η) and the pheromone trails (τ) as shown in Equation 9. At each step of the procedure we have to specify the set of allowed points. The following criteria have to be met:

  a) The production rule, after the term is added, covers more cases than a user-defined parameter *min_cases_per_rule.*

  b) The production rule contains only a single term that includes a certain predictor attribute to avoid an invalid rule antecedent such as "temperature = low AND temperature = high".

  The path construction procedure stops when the set of allowed points is empty and the path is considered as finished.

  The *min_cases_per_rule* parameter value is set to 5 [18].

- Heuristic function. The heuristic function η that describes the local quality of the environment is based on the entropy [30]. The same approach it is used in one of the competing classification algorithms, C4.5 [8].

- Solution quality formula that we use to evaluate the complete ant path:

$$Q = sensitivity * specificity \tag{12}$$

where the sensitivity and specificity are computed for a completed ant path that represents single production rule over a current training set, i.e., the training cases that are used to extract production rule with the ACO optimization algorithm.

- Pheromone update formula:

$$\tau_{mn}(t+1) = \frac{\tau_{mn}(t) + Q\tau_{mn}(t)}{\sum_{\forall mn \in R} \tau_{mn}} \tag{13}$$

where t is iteration number, m , n are points in search space, Q is the quality of candidate solution, and R is a set of trails that constitute the candidate solution.

- The ACO algorithm stop condition. The search procedure that extracts single production rule stops when the ant colony converges or the maximum number of ants constructs the complete path. The algorithm converges when the number of successive ants that generate the same candidate solution is equal to the value of *no_rules_converge* parameter.

  The *no_rules_converge* parameter value is set to 10 [18] and maximum number of ants equals 3000 [18].

- Additional parameters. The ACO algorithm contains a few additional parameters that have to be specified, namely the transition probability parameters α and β. The transition probability defined in Equation 9 characterizes the choice of trails that could be added to the candidate solution. We use the following values: $\alpha = \beta = 1$ [18]; this means that both the pheromone trails and the heuristic function have equal impact on the value of the transition probability.

In the following subsections we discuss stop conditions of the AntMiner classification algorithm. The rule extraction procedure stops when the current training set contains fewer cases than an user-defined parameter *max_uncovered_cases*. The *max_uncovered_cases* parameter value is set to 10 [18].

3.2.1 Rule representation and rule encoding

AntMiner algorithm extracts a set of production rules. Single production rule contains a number of terms, where each term can contain only single logical test. This means that rule antecedent is a conjunction of logical tests, see Figure 9. In the AntMiner algorithm each term is a single logical test that involves one attribute. This is a simplification of the rule representation shown in Figure 1.

```
IF
    attribute_1 = value_1
AND
    attribute_2 = value_2
AND
    ...
AND
    attribute_N = value_N
THEN class = value
```

Figure 9: Simplified production rule representation in AntMiner algorithm, where N is the number of attributes.

An encoding scheme that translates a completed ant path into relevant production rule is not needed as each ant path represents a number of terms that are present in the production rule, which is the desired representation. The path construction procedure is explained in the previous section.

3.2.2 Rule extraction procedure

AntMiner rule extraction procedure creates a set of production rules that can be used to assign class label to previously unknown data cases. Rules are created one after another and in each iteration single production rule is generated. While class attribute is not subject to optimization, its value is set only after the rule antecedent is completed (single ant constructed the complete path between source and destination point); it is unknown to this point of rule extraction procedure. We select a majority class label among the covered cases in order to maximize the quality of the rule.

The main algorithm, see Figure 10, starts with initialization of a current training set and is divided into a set phases. In the first, core step, ACO algorithm is used to extract a set of candidate production rules. Only the best rule is chosen and added to the output ruleset. Next, the selected production rule is pruned to remove irrelevant terms to avoid potential overfitting to the current training set. Finally, all training cases correctly classified by the generated rule are removed from the current training set and next algorithm iteration starts. Rule extraction algorithm stops when the number of remaining training cases is smaller that user-defined threshold value.

```
initialize(aco, data, ruleset)
WHILE number_of_training_cases > max_number_of_uncovered_cases
    rule = generate_rule_with_ACO(aco, data)
    update_ruleset(ruleset, rule)
    update_training_dataset(data, rule)
END
post_process_ruleset(ruleset, data)
```

Figure 10: Pseudo-code of the AntMiner rule extraction algorithm

When the rule extraction procedure stops and the ruleset contains a set of rules generated from training set, a default rule that covers remaining training cases is created and concatenated to the ruleset. Default rule is in the form "**IF** *TRUE* **THEN** class=*default_value*", where *default_value* is the most common class label in the remaining training cases.

3.2.3 Classification procedure

Generated ruleset, which was extracted by the AntMiner algorithm, can be used to classify a new case that was unseen during training process. The discovered production rules are applied in the order in which they were extracted and stored in the ruleset. The first rule that covers the new case is used and the case is assigned the class indicated by the corresponding rule consequent.

The ruleset contains a default rule, which is used for all new cases that are not covered by any of the production rules extracted during rule extraction procedure by the ACO algorithm.

3.3 Summary

In this chapter we discussed the current biologically inspired classification algorithms such as PSO II and AntMiner and provided details about the parametrization of the considered algorithms. We described several aspects of the algorithms such as the rule representation and encoding (i.e., defines the mapping between the optimization algorithm and the production rules), rule extraction procedure (i.e., the method used to extract production rules from the training set), and classification procedure (i.e., the method to assign class label to test case using the extracted production rules).

4 Proposed biologically inspired classification algorithm

In this chapter we discuss the proposed biologically inspired classification algorithm. The discussion is organized in the similar fashion as the description of the PSO II and AntMiner algorithms to allow for seamless comparison between the main concepts of the considered biologically inspired classification algorithms. The last section contains summarized characteristics of the above approaches.

4.1 Motivation

The current biologically inspired classification algorithms such as PSO II and AntMiner focus on the accuracy as the main goal in the design of the corresponding classification algorithms. The algorithms are characterized by the straightforward production rule representation and the classification procedure that resembles the approach used by the decision list classifiers.

In this work we extend the production rule representation and the rule extraction procedures to design accurate and modular classifier. We acknowledge that the classification accuracy is an important aspect of the classification algorithm. To this end, we use an extensive experimental environment to compare the accuracy of models generated by the proposed method with the accuracies of the considered biologically inspired classification algorithms and other traditional "white-box" classification algorithms such as RIPPER, C4.5, and PART. Additionally, the proposed algorithm is characterized by modularity of the generated model (i.e., the extracted production rules are independent of each other and thus they can be used separately). This feature separates our method from the current biologically inspired classification algorithms that generate a set of dependent production rules. Finally, we analyze and quantify the completeness of the classification model, i.e., in the case of the proposed method the ruleset contains production rules for each class present in the training set. This is again the defining characteristics of our method, while the current biologically inspired classification algorithms may provide incomplete model. Our experiment measure how often the competing methods (including biologically inspired classification algorithms and the other considered "white-box" methods) generate incomplete models.

4.2 Proposed method

In this section we discuss the parametrization and the design of the proposed biologically inspired classification algorithm. The next few subsections contain description of several aspects related to the classification model such as rule representation, rule extraction procedure, and classification procedure.

The proposed biologically inspired classification algorithm uses the BPSO algorithm to discover a set of production rules that constitute the classification model. The algorithm defines following aspects of the underlying BPSO algorithm:

- Particle quality formula used to evaluate current particle position

$$Q(x) = sensitivity * specificity \tag{14}$$

where x is the particle position and the sensitivity and specificity are computed for a given particle that represents a single production rule over a current training set, i.e., the training cases that are used to extract the production rule with the BPSO optimization algorithm.

The above particle quality formula favours general production rules (i.e., the production rules that cover larger number of the positive cases) that at same time minimize the number of covered negative cases. The ideal production rule, in terms of the proposed evaluation measurement, covers all positive cases and no negative case. This quality formula was proposed in [31][32] and is also used in the competing biologically inspired classification algorithms [17][18].

- Particle movement equation parameters. We analyze and optimize the parameters to find the best performing combination of the particle movement parameters X, φ_1, and φ_2. The detailed analysis is provided in the experimental section of this work.
- The PSO algorithm stop condition. In the proposed method we specify a fixed number of iterations that are performed during the optimization procedure. The value we choose is big enough to assure the stagnation of the BPSO optimization procedure during the rule extraction procedure and equals 1000.

We analyze all benchmarking problems that are used in the experimental study, see Chapter 5, to ensure that the value of quality function does not increase after the selected number of the BPSO algorithm iterations during the rule extraction procedure.

In the following subsections we discuss the stop condition of the proposed classification algorithm. The rule extraction procedure stops when the current training set contains fewer cases than user-defined parameter The value is set to 5% of cases in the positive class of the current training set.

4.2.1 Rule representation and rule encoding

The proposed biologically inspired classification algorithm extracts a set o prediction rules. Single prediction rule antecedent is composed of the number of terms connected by the logical conjunction. Each term is a disjunction of logical tests over the single predictor attribute. The rule consequent contains a predicted class label. The rule representation is shown in Figure 1.

An encoding scheme that translates the BPSO particle (each particle is a binary string of length D) into the relevant production rule is necessary to associate the problem of extracting production rules with the BPSO optimization algorithm. In the proposed algorithm D is the total number of attribute values, computed as a sum of the number of unique values of all predictor attributes. The BPSO particle position bit string is divided into a number of substrings. Single substring corresponds to the certain predictor attribute and represents one term, e.g., if the number of predictor attributes is N then the bit string contains N substrings and if the predictor attribute has N_i unique values then the corresponding substring contains N_i bits. Single substring bit corresponds to the one logical test from the term, i.e., if nominal attribute A has three unique values {*small*, *medium*, *large*} then the substring contains three bits and a pattern

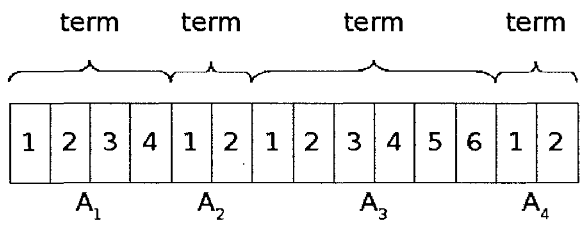


Figure 11: The BPSO particle position bit string (14 bits length) encoding for the production rule with four terms (attributes A_1, A_2, A_3, A_4). Each bit corresponds to the certain value of the specific attribute.

011 is read as term $A = \textit{medium}$ OR $A = \textit{large}$. While the order of the predictor attribute values is not important during optimization procedure, we specify the order of the values in the BPSO particle position bit string to correctly encode the production rule, see Figure 11.

The encoding scheme has two special cases, when all bits are set to one and when all bits are set to zero. When all bits are set to one, the rule antecedent covers all cases and the production rule can be interpreted as the default rule. Similarly, when all bits that correspond to a certain term are set to one then the attribute is not included in the rule antecedent. When all bits are set to zero, the rule antecedent does not cover any case and the production rule is treated as invalid. The interpretation is alike when all bits that correspond to the single term are set to zero.

We reuse the encoding scheme that was introduced in [33][34] and which was used used in biologically inspired classification algorithms that extract the production rules with the GA algorithm [16].

4.2.2 Rule extraction procedure

The proposed rule procedure algorithm iteratively generates production rules from the input training dataset. A specific class label is chosen in advance for each generated rule, which happens before the actual procedure of single rule generation starts. The algorithm is designed to create at least a single rule for each class label that is available in the training set; this means that we explicitly set a minimum number of rules that would be present in the extracted ruleset.

The main algorithm, see Figure 12 is divided into inter and outer iterations. In the former part we only select a current positive class label and initialize the rule extraction procedure. The letter part is the actual rule extraction procedure in which one production rule is generated in a single iteration. In each outer iteration we execute a number of inter iterations, thus generating a number of production rules for the same class; we refer this set of production rules a submodel. We make this distinction because all rules in the submodel share the same class label and no other rule consequent extracted during complete run of the rule extraction algorithm contains this class label. Generated ruleset does not contain any default rule and is composed of the set of submodels that cover all class labels that are available in the training set.

```
initialize(ruleset, bpso, data)
FOR class_label IN set_of_available_class_labels
    initialize_data(data, class_label)
    WHILE number_of_training_cases > max_number_of_uncovered_cases
        rule = generate_rule_with_BPSO(bpso, data, class_label)
        update_ruleset(ruleset, rule)
        update_training_dataset(data, rule)
    END
END
```

Figure 12: Pseudo-code of the proposed rule extraction algorithm

Outer loop of the rule extraction algorithm starts with the training set reinitialization; all submodels are generated from the complete training set. We select a unique positive class label and divide the dataset into *positive* and *negative* parts. The outer loop stops when all class labels from the training set are processed.

Inner loop contains actual rule extraction procedure; in each iteration a single production rule is generated and added to the selected submodel. In the first, core step, the BPSO algorithm is used to find the best production rule that can be extracted from the current training set. The extracted production rules is not modified, i.e., we do not perform the rule pruning, as in the scope of this work we do not analyze size of the production rules. Next, the generated production rule is added to the selected submodel with the corresponding quality value as computed during the BPSO optimization procedure. Finally, we modify the training set by removing all correctly classified cases. The inner loop stop when the number of the remaining training cases in the positive part is smaller than a user-defined threshold value.

When compared with the existing biologically inspired algorithms the distinctive feature of the proposed method is the two level design of our approach. As the result of this design, we create as set of submodels that consist of modular production rules, i.e., each rule can be interpreted and used independently of the remaining ruleset. This is in contrast to the rules generated by the previously described PSO II and AntMiner biologically inspired classification algorithms. While the extracted model is more flexible and provides at least one rule for each class label present in the training set, the main disadvantage of this approach is the increased computational cost; each submodel is generated from the complete training set.

4.2.3 Classification procedure

The generated ruleset can be used to predict class label of the unknown cases, i.e., cases that are not included in the training set. In the proposed biologically inspired classification algorithm the production rules are modular and can be used independently. At the same time, the complete extracted model should be used to perform classification ,i.e., one rule can only predict one class label, while the entire ruleset allows for classification into multiple classes.

Classification procedure ranks the generated submodels according to the number of production rules that covers the test case. The submodel with the highest quality, computed as the sum of quality values of rules, within the submodel, that cover the test case, constitutes the prediction. However, the test case may not be covered by any of the generated production rules. In this situation the prediction is based on the "most compatible" production rules, i.e., production rules with the biggest number of passed terms. As more than one production rule may by "compatible" at the same level, i.e., with exactly the same number of the passed terms, we use all the selected production rules to rank submodels according to the sum of production rule quality values.

When compared with the existing biologically inspired algorithms the main difference is that we do not use the generated ruleset in any specific order as the rules are modular and independent. Additionally, instead of using the default rule to provide prediction for the test cases that are not covered by any of the extracted production rules we propose a novel procedure to provide output class label in this situation.

One distinctive advantage of the proposed approach is the submodel ranking. During classification procedure we compute a quality measure for each of the submodels. This value is used to rank submodels, and the best is used for the prediction. While the other considered methods provide only the predicted class label, in our case we provide the user additional feedback information in terms of the confidence value that is assigned to each of the submodels. The confidence value can be used to compare different submodels, however we lose this information when using only single prediction rule from the extracted ruleset.

4.2.4 Comparison with existing biologically inspired algorithms

Table 2 presents a comparison between the proposed method and existing biologically inspired algorithms such as PSO II and AntMiner. The Table contains a condensed overview of characteristics of the methods analyzed in this work. The comparison is organized in similar fashion as the the descriptions of the considered three biologically inspired classification algorithms and is separated into three main design steps.

Main design steps	**Detailed characteristics**	**Algorithm**		
		PSO II	**AntMiner**	**Proposed method**
Rule representation and encoding	Rule can contain nominal attributes	Yes	Yes	Yes
	Rule can contain numerical attributes	Yes, but never tested	No	No
	Encoding assumes order of the attribute values	Yes	No	No
Rule extraction procedure	Ruleset contains default rule	Yes	Yes	No
	Ruleset contains rules for each class	No	No	Yes
	Rules are modular and can be used independently	No	No	Yes
	Rules with same consequent are grouped together	No	No	Yes
	Rules are pruned before adding to the ruleset	Yes	Yes	No
Classification procedure	The rules are used in the extraction order	Yes	Yes	No
	Classification method provides prediction for all possible predictor attributes values	Yes	Yes	Yes
	The models includes additional feedback information i.e., confidence factor	No	No	Yes
Overall	Algorithm works with unknown cases, i.e., non-missing values of unknown cases are used to generate model.	No	Yes	Yes

Table 2: Comparison of the proposed method with the existing biologically inspired algorithms

4.3 Summary

In this chapter we described the proposed biologically inspired classification algorithm. We presented several aspects of the proposed method such as the parametrization, the rule representation and encoding (i.e., defines the mapping between the optimization algorithm and the production rules), the rule extraction procedure (i.e., the method used to extract production rules from the training set), and the classification procedure (i.e., the method to assign class label to test case using the extracted production rules) and contrasted the proposed design with the other biologically inspired classification algorithms such as PSO II and AntMiner. As the result of the several extensions to the considered algorithms the proposed method extracts a complete set of modular production rules.

5 Experimental results

In this chapter we experimentally compare the proposed biologically inspired classification method with the existing "white-box" classification methods, such as RIPPER, C4.5, PART, and the current biologically inspired classification algorithms such as PSO II and AntMiner. First, we describe the datasets used in the experiments, including several real world datasets and two synthetic datasets. Next, we explain the parametrization of the proposed method. Finally, we provide the results of the comparison between all the six learners.

5.1 Synthetic datasets

To evaluate the proposed biologically inspired classification method and compare it with the considered two biologically inspired algorithms we perform several experiments on the two synthetic datasets. We apply the three biologically inspired classification algorithms to extract the classification models from the two synthetic datasets and contrast the rule extraction procedures and output rulesets.

Both synthetic datasets are multi-class and contain three unique class attribute values (i.e., class 1, class 2, and class 3). The predictor attributes (i.e., attribute 1 and attribute 2, referred to as A1 and A2, respectively) are defined as follows: $A1 \in \{0,1,2,3\}$, $A2 \in \{0,1,2\}$. Figure 13 shows the generator cases that are used to create the synthetic datasets; each class contains a specific number of training cases that are randomly chosen from the generator cases.

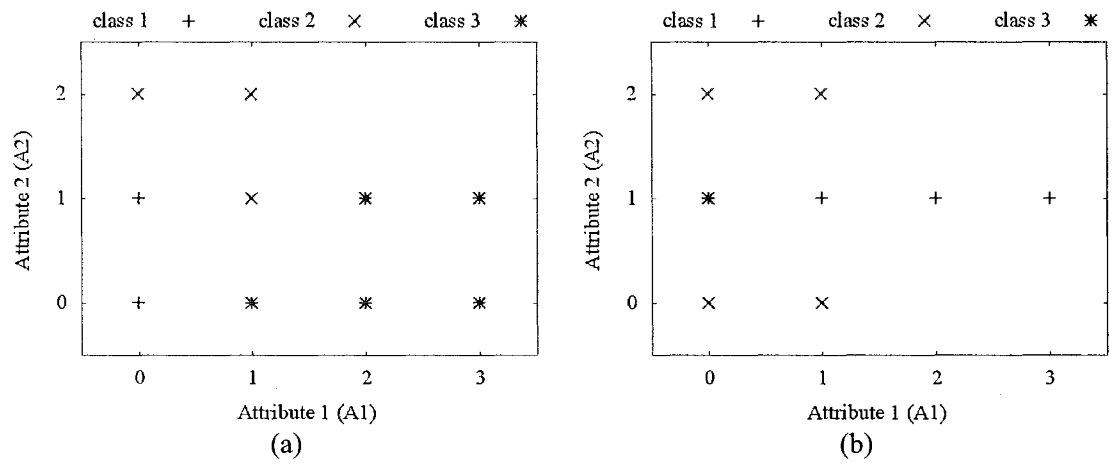


Figure 13: The generator cases for the multi class synthetic datasets, i.e., the training synthetic dataset contains random number of the generator cases. The class 1 generator cases are indicated as "+", the class 2 generator cases are indicated as "x", and the class 3 generator cases are indicated as "*". Panel (a) defines generator cases for dataset 1, while Panel (b) shows generator cases for dataset 2. The x-axis and y-axis correspond to the values of attribute 1 (A1) and 2 (A2), respectively.

The number of the cases for each unique class label in the synthetic dataset 1 is as follows: class 1 contains 320 cases, class 2 contains 80 cases, and class 3 contains 10 cases.

The number of the cases for each unique class label in the synthetic dataset 2 is as follows: class 1 contains 10 cases, class 2 contains 60 cases, and class 3 contains 160 cases.

The uneven distribution of the cases in the both multi-class datasets is used to expose advantages and disadvantages of the considered biologically inspired classification algorithms.

In the experiments that concern synthetic datasets we extract the classification model and evaluate it on the same, complete dataset. While the quality of the generated classification model may be misleading, the main goal of the experiments is to compare the advantages and disadvantages of the rule extraction algorithms used in PSO II, AntMiner, and the proposed classification method.

5.2 Benchmarking datasets

To evaluate the proposed biologically inspired classification method and compare it with considered classification algorithms we perform experiments with several real word datasets, also known as benchmarking problems. We apply the selected six classification algorithms to extract the classification models from the datasets.

The real world datasets are publicly available and they were retrieved from the UCI repository [35]. The selected datasets, see Table 3, provide a comprehensive environment to test performance of the proposed and the competing methods. The benchmarking problems range from 101 to 12960 cases, the number of attributes is between 7 and 35, and the number of class labels ranges between 2 and 7. The datasets include both binary and multi class problems.

#	Abbr.	Full name	Number of cases	Number of class labels	Distribution of cases in classes	Number of attributes
1	bc	breast cancer	277	2	196/81	10
2	bcw	breast cancer Wisconsin	683	2	444/239	10
3	car	car evaluation	1728	4	1210/384/69/65	7
4	cmc	contraceptive method	1473	3	629/333/511	10
5	derm	dermatology	358	6	111/60/71/48/48/20	35
6	dna	dna	3186	3	767/765/1654	61
7	lymph	lymphography	148	4	2/81/61/4	19
8	mush	mushroom	5644	2	3488/2156	23
9	nurse	nursery	12960	5	4320/3/328/4266/4044	9
10	vote	congressional voting records	232	2	124/108	17
11	zoo	zoo	101	7	41/20/5/13/4/8/10	17

Table 3: Properties of the benchmarking datasets. The number of attributes includes both the predictor attributes and the class attribute.

5.3 Parametrization of the proposed method

The proposed biologically inspired algorithm contains a number of parameters that have to be specified. In this section we discuss the parameter selection procedure for the BPSO algorithm, which is used to extract production rules in the proposed classification algorithm. The PSO algorithm parametrization proposed in [28][29] guarantees the convergence of the swarm and provides an excellent choice for the parameters. However the BPSO algorithm lacks such analysis and we parametrize the parameters such as X, φ_1, φ_2, and the neighbourhood model. We also discuss other parameters such as stop condition parameters, that are required in the proposed classification algorithm.

We separate the required parameters into two groups:

- BPSO algorithm swarm behaviour parameters. This group contains all parameters that have direct impact on the swarm search procedure, i.e., movement, information sharing, and the swarm size. We analyze these parameters in the proposed parametrization procedure.
- Stop condition parameters. This group includes two parameters, the number of iterations of the BPSO optimization algorithm, and the minimal size of the current training set during the rule extraction procedure. These parameters were already discussed and specified in Chapter 4

The parameter selection procedure for the BPSO algorithm focuses on the BPSO particle movement equation, see Equation 8. However, we also analyze different neighbourhood models and swarm sizes. We use only a subset of the benchmarking problems listed in Table 3, which are characterized by the small size, i.e., the number of cases, due to computationally expensive nature of the parametrization procedure. We include the following datasets: bc, bcw, derm, lymph, vote, and zoo.

The following parameters are optimized:

- Particle movement equation parameters. We analyze the X, φ_1, and φ_2 parameters of the BPSO particle movement equation. In the analysis $\varphi_1 = \varphi_2$, this means that the private thinking and the social behaviour have equal impact on the particle movement throughout the search space. We execute a grid search over the following sets of values $X = \{0.75, 1.00, 1.25\}$ and $\varphi = \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$. The default settings for the BPSO algorithm particle movement parameters suggested in the [23] is as follows $X=1.00$ and $\varphi_1=\varphi_2=2.0$

- Neighbourhood model. We use the global (referred to as global) and closest index (referred to as index) based neighbourhood model. In the latter case only two closest particles are considered. The default setting for the BPSO algorithm neighbourhood model, which was suggested in the [23], is global.
- Swarm size. We check the trade-off between the quality of the classification model and the computational costs to generate the model with the BPSO optimization algorithm. We use the following sizes: 25, 50, and 100.

In order to select the best combination of the parameters for the BPSO algorithm we analyze both the accuracy of the generated model and the number of production rules in the generated ruleset. Both values are combined into a single quality measure that is defined as follows:

$$Q_p=(100-accuracy)*\#\,rules \qquad (15)$$

The goal of the parameter selection procedure is to find the combination of above parameters that will result with smallest value of the quality measure defined in Equation 15. However in the discussion we also consider the trade-off between the obtained values of the quality measure and the computational costs, i.e., bigger size of the swarm will increase the time requirements.

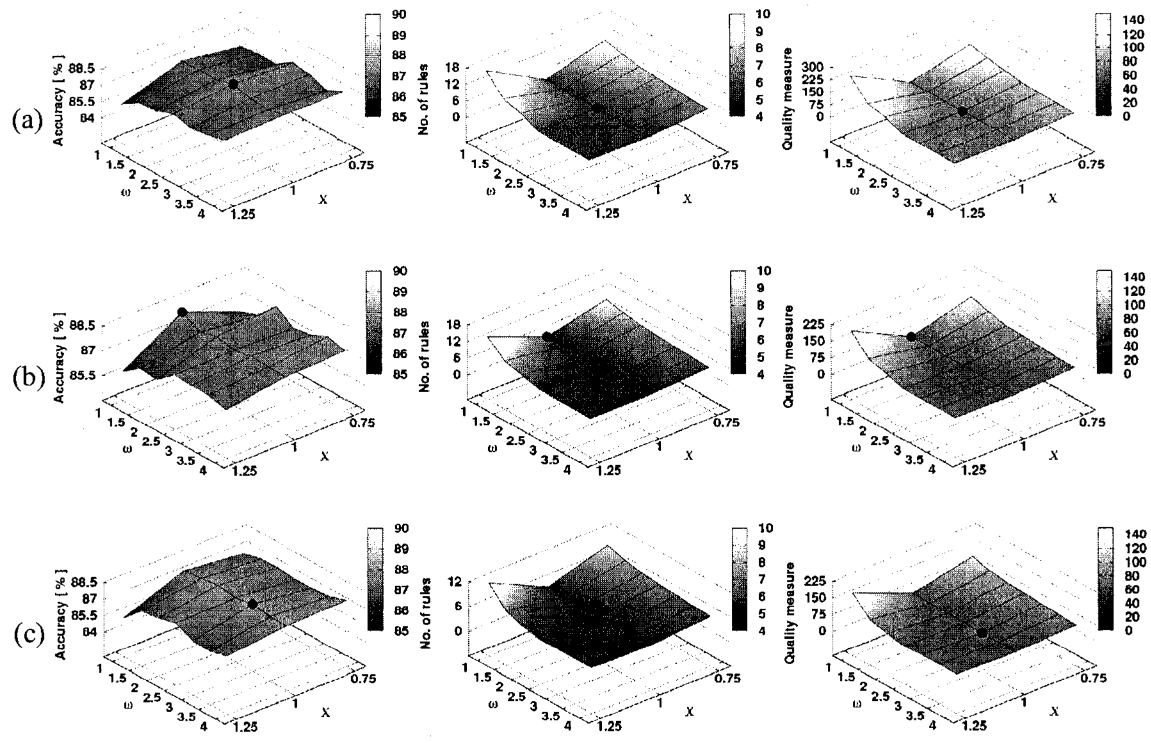


Figure 14: Visualization of the grid parametrization results for the global neighbourhood model with (a) 25, (b) 50, and (c) 100 particles in swarm. The plots present the accuracy of the generated model (left column), the number of rules in the extracted model (central column) and the quality measure Q_p (right column) over the selected values of the BPSO particle movement equation parameters X and φ. The point with the best value of Q_p is indicated as black dot. The results are based on the 10-fold cross-validation tests that were repeated five times over the six selected benchmarking problems.

The visualization of the parametrization results is in Figure 14 and Figure 15, which show results for global and index neighbourhood models, respectively. Each plot contains a summary of the results for a certain neighbourhood model and swarm size over the selected values of the BPSO particle movement equation parameters, X and φ. Single point on the grid in a given plot is an average obtained from the 10-fold cross-validation procedure that is repeated five times over the six selected benchmarking problems. We repeat the 10-fold cross-validation because the proposed biologically inspired classification algorithm is nondeterministic, i.e., pseudo-random numbers are used during the initialization and optimization.

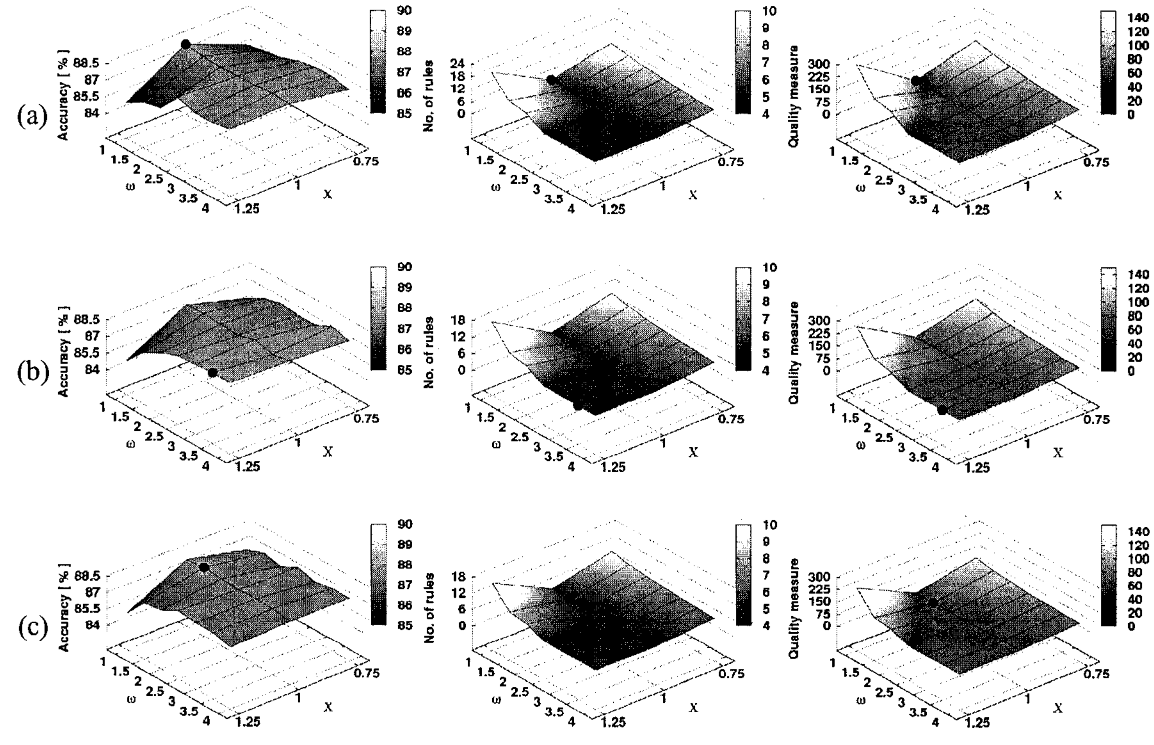


Figure 15: Visualization of the grid parametrization results for the index neighbourhood model with (a) 25, (b) 50, and (c) 100 particles in swarm. The plots present the accuracy of the generated model (left column), the number of rules in the extracted model (central column) and the quality measure Q_p (right column) over the selected values of the BPSO particle movement equation parameters X and φ . The point with the best value of Q_p is indicated as black dot. The results are based on the 10-fold cross-validation tests that were repeated five times over the six selected benchmarking problems.

We summarize the results of the parameter selection procedure in Table 4. We report the minimal Q_p values for each neighbourhood model and swarm size together with the corresponding number of rules in the extracted model and the mean accuracy over the six selected benchmarking problems. Additionally we provide the results for the default parameters suggested in [23], i.e., $X=1.00$, $\varphi_1=\varphi_2=2.0$, and global neighbourhood model; these results are underscored in Table 4.

Parameters				**Results**			
Neighbourhood model	**Number of particles in swarm**	X	φ	**Accuracy [%]**	**Number of rules**	Q_p **value**	**Runtime [sec]**
global	25	1.00	2.50	87.58	5.25	65.23	848
<u>global</u>	<u>25</u>	<u>1.00</u>	<u>2.00</u>	<u>87.10</u>	<u>5.24</u>	<u>67.62</u>	<u>811</u>
global	50	1.00	1.00	87.86	5.09	61.82	1650
<u>global</u>	<u>50</u>	<u>1.00</u>	<u>2.00</u>	<u>87.58</u>	<u>5.11</u>	<u>63.52</u>	<u>1661</u>
<u>global</u>	<u>100</u>	<u>1.00</u>	<u>2.00</u>	<u>87.86</u>	<u>5.03</u>	<u>61.09</u>	<u>3291</u>
global	100	1.00	3.00	87.90	5.04	61.01	3196
index	**25**	**1.00**	**1.00**	**88.06**	**5.02**	**59.99**	**799**
index	50	1.25	3.50	88.16	5.00	59.25	1593
index	100	1.00	1.50	88.04	5.00	59.80	3234

Table 4: Listing of the best combinations of X and φ parameters for each considered combination of the neighbourhood model and number of particles in swarm. The results are based on the 10-fold cross-validation tests that were repeated five times over the six selected benchmarking problems. The runtime is the time needed to extract all classification models for the six considered benchmarking problems in the 10-fold cross validation tests. Selected parameters are shown in bold. The default parameters as suggested in [23] are underlined.

The results show that the accuracy difference between the best and the worst combinations of BPSO parameters is very small and equals $\sim 0.5\%$, when we do not include the default parameters. The difference in the case of the number of rules in the extracted classification model equals 0.25.

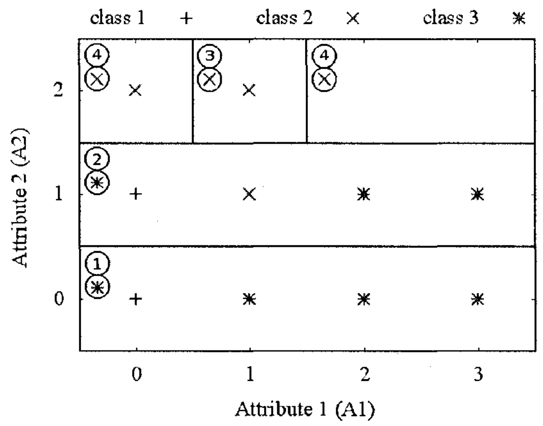
We select $X=1.00$, $\varphi_1=\varphi_2=1.0$ and index based neighbourhood model to implement the proposed biologically inspired algorithm. We use these values in all subsequent experiments. While a few other parameter sets can lead to slightly improved quality (the value of Q_p) the selected parameters provides the best trade-off between the computational costs, measured using the running time, and the obtained quality.

5.4 Results on the synthetic dataset

In this section we provide and discuss the results of the comparison between the three selected learners, PSO II, AntMiner, and the proposed biologically inspired classification algorithm on the two synthetic datasets. We describe the extracted classification models in the context of the design of the considered classification algorithms.

The two considered biologically inspired classification algorithms, AntMiner and the proposed method, create classification models that correctly classify all the training cases for both synthetic datasets. However, the PSO II algorithm accuracy is lower than 100% on both considered synthetic datasets.

Figure 16 shows the decision boundaries derived from the generated classification models and the production rules that constitute the classification model extracted from the synthetic dataset 1. The order of the rules is important for the classification procedure of the PSO II and AntMiner algorithms.



PSO II production rules:

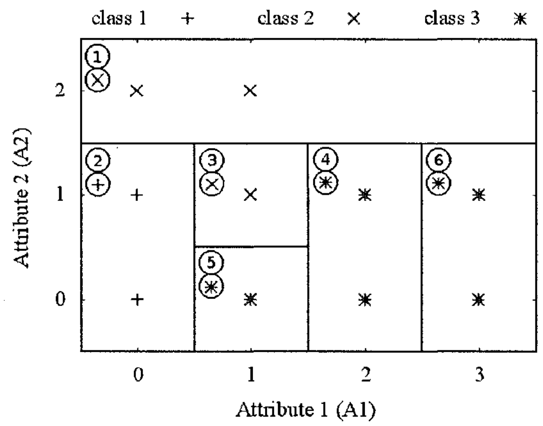
(1) IF A2 = 0 THEN class = 3

(2) IF A2 = 1 THEN class = 3

(3) IF A1 = 1 THEN class = 2

(4) IF A2 = 2 THEN class = 2

(5) IF *TRUE* THEN class = 1



AntMiner production rules:

(1) IF A2 = 2 THEN class = 2

(2) IF A1 = 0 THEN class = 1

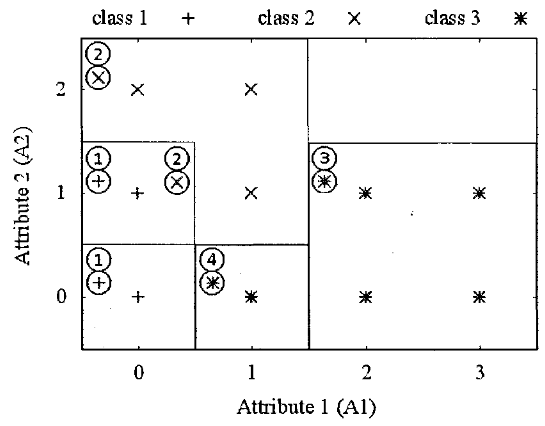
(3) IF A1 = 1 AND A2 = 1 THEN class = 2

(4) IF A1 = 2 THEN class = 3

(5) IF A1 = 1 THEN class = 3

(6) IF A1 = 3 THEN class = 3

(7) IF *TRUE* THEN class = 1



The proposed method production rules:

(1) IF A1 = 0 AND (A2 = 0 OR A2 = 1) THEN class = 1

(2) IF (A1 = 0 OR A1 = 1) AND (A2 = 0 OR A2 = 1) THEN class = 2

(3) IF (A1 = 2 OR A1 = 3) AND (A2 = 0 OR A2 = 1) THEN class = 3

(4) IF A1 = 1 AND A2 = 0 THEN class = 3

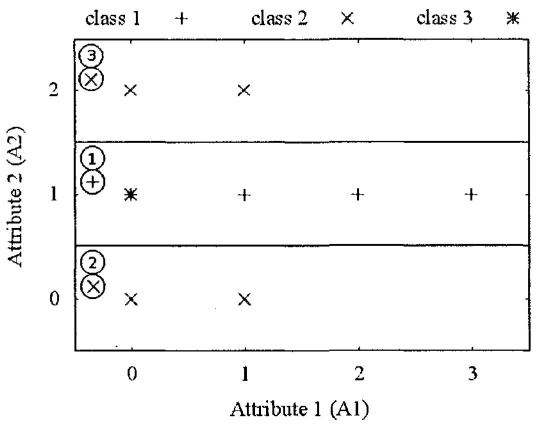
Figure 16: Visualization of the decision boundaries derived from the generated classification models and the production rules that constitute the classification model extracted from the synthetic dataset 1 by PSO II (top row), AntMiner (middle row), and the proposed classification algorithm (bottom row). The decision regions and the corresponding production rules are numbered to show the order in which they were generated, which is important for the PSO II and AntMiner algorithms. Each number on the decision region is associated with the rule consequent shown in the circle below the number. Generator cases are indicates as "+", "x", and "*".

PSO II classification model (top row) contains five production rules, including the default rule. Rules (1) and (2) assign the cases to the majority class in the synthetic dataset 1, which is the class 3. The possible misclassification of the test cases is the result of the quality function that is used in the PSO II algorithm to evaluate the production rules and the specific rule representation. Rule (3) correctly classifies the cases in the class 2. However, because of the classification procedure used in PSO II, it only covers one generator case (the production rules have to be applied in the order in which they were extracted from the training set). This means that the rule (3) cannot be applied before the rules (1) and (2) that were extracted earlier. The generated production rules (1), (2), (3), and (4) cover all possible values of attributes A1 and A2. This means that the default rule will not be applied and while the default rule consequent points to the class 1, the test cases will not be classified as class 1.

AntMiner classification model (middle row) contains seven production rules, including the default rule. The extracted production rules correctly classify all the generator cases. The rule extraction procedure used by AntMiner makes the order of the classes in the production rules arbitrary, e.g., the rules (1) and (2) for the class 2 are separated by the rule (2) with the class 1. The shape of the decision boundaries reveals underlying complexity of the classification with the extracted production rules, which is is in contrast to the simplicity of the ruleset itself. The complexity of the decision boundaries stands from applying the classification procedure on the extracted ruleset. Similarly as in the case of the PSO II algorithm, the complete set of the extracted production rules covers all possible values of attributes A1 and A2. This means that the default rule will not be applied.

The proposed method classification model (bottom row) contains four production rules. The complete classification model correctly classifies the generator cases. However, the individual production rules may incorrectly classify the test cases. In the extracted classification model the rules (1) and (2) overlap, and the classification decision is based on the quality of the production rules measured as described in Chapter 4. The production rules extracted in the proposed classification algorithm contain a complete description of the decision boundary and can be used separately, which is in contrast to the competing biologically inspired classification algorithms. This difference is result of the design of the rule extraction procedure. The extracted production rules do not cover all possible values of attributes A1 and A2. While the synthetic dataset does not contain any generator cases for those regions, the proposed classification algorithm will still classify the test cases in those regions. The decision is based on the classification procedure described in Section 4.2.3.

Figure 17 shows the decision boundaries derived from the generated classification models and the production rules that constitute the classification model extracted from the synthetic dataset 2. The order of the rules is important for the classification procedure of the PSO II and AntMiner algorithms.



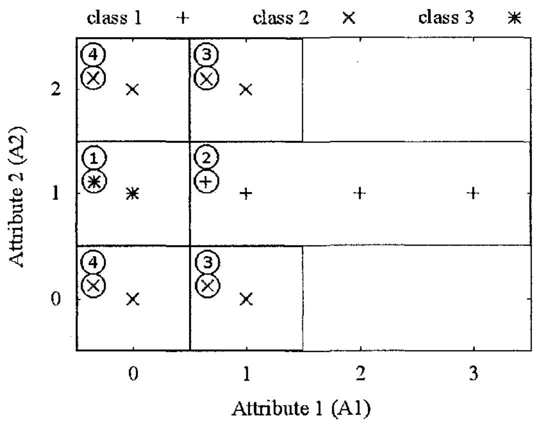
PSO II production rules:

(1) IF A2 = 1 THEN class = 1

(2) IF A2 = 0 THEN class = 2

(3) IF A2 = 2 THEN class = 2

(4) IF *TRUE* THEN class = 3



AntMiner production rules:

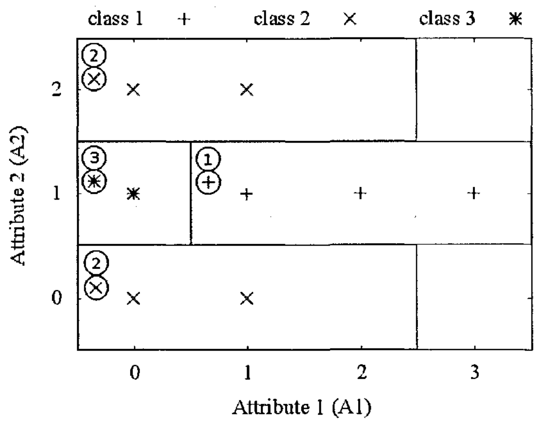
(1) IF A1 = 0 AND A2 = 1 THEN class = 3

(2) IF A2 = 1 THEN class = 1

(3) IF A1 = 1 THEN class = 2

(4) IF A1 = 0 THEN class = 2

(5) IF *TRUE* THEN class = 1



The proposed method production rules:

(1) IF (A1 = 1 OR A1 = 2 OR A1 = 3) AND A2 = 1 THEN class = 1

(2) IF (A1 = 0 OR A1 = 1 OR A1 = 2) AND (A2 = 0 OR A2 = 2) THEN class = 2

(3) IF A1 = 0 AND A2 = 1 THEN class = 3

Figure 17: Visualization of the decision boundaries derived from the generated classification models and the production rules that constitute the classification model extracted from the synthetic dataset 1 by PSO II (top row), AntMiner (middle row), and the proposed classification algorithm (bottom row). The decision regions and the corresponding production rules are numbered to show the order in which they were generated, which is important for the PSO II and AntMiner algorithms. Each number on the decision region is associated with the rule consequent shown in the circle below the number. Generator cases are indicates as "+", "x", and "*".

PSO II classification model (top row) contains four production rules, including the default rule. Rule (1) assign the cases to the majority class in the synthetic dataset 2, which is the class 1. The possible misclassification of the test cases is the result of the quality function that is used in the PSO II algorithm to evaluate the production rules and the specific rule representation. The extracted production rules (1), (2), and (3) cover all possible values of the attributes A1 and A2. This means that the default rule will not be applied and while the default rule consequent contains the class 3, the test cases will not be classified as class 3.

AntMiner classification model (middle row) contains five production rules, including the default rule. All extracted production rules correctly classify the generator cases. Similarly as in the case of synthetic dataset 1, we observe that the decision boundaries are relatively complex. The extracted production rules (1), (2), (3), and (4) do not cover all possible values of the attributes A1 and A2. This means that the default rule will be applied for these regions and test cases will be classified as class 1.

The proposed method classification model (bottom row) contains three production rules. Both the complete classification model and each individual production rules correctly classify the generator cases. We also note that shape of the decision boundaries may be different in consecutive runs of the proposed rule extraction procedure due to the randomness of the search procedure, i.e., rule (2) could also include points (A1=3, A2=0) and (A1=3, A2=2). We note that the proposed method can generate production rules that describe cases tat were not included in the training set. In particular, rule (2) covers points (A1=2, A2=0) and (A1=2, A2=2), which do not correspond to the generator cases. The presence of logical tests A1=2 and A1=3 in rule (2) is arbitrary. This is the result of selection performed with the quality function used in the proposed classification algorithm to evaluate the production rules during the optimization procedure.

5.5 Results on the benchmarking datasets

In this section we provide the results of the comparison between the six selected learners, RIPPER, C4.5, PART, PSO II, AntMiner, and the proposed biologically inspired classification algorithm. The experiments are performed on the selected eleven real world benchmarking problems.

We use RIPPER, C4.5 and PART classification algorithms implementations from the Weka suite [36]. The biologically inspired algorithms, PSO II [17] and AntMiner [18], were obtained from the authors.

In the experimental study we use the 10-fold cross validation procedure to compare the six learners. Similarly as in the synthetic datasets comparison, we repeat the 10-fold cross-validation five times for PSO II, AntMiner and the proposed classification method. The reported quality of the generated model is an average over the ten folds together with the corresponding standard deviation. In the case of PSO II, AntMiner, and the proposed classifier, where 10-fold cross validation is repeated five times, we first average the measurements for the same folds obtained in the multiple runs, and then these averages are used to compute average and standard deviation for the ten folds.

We separate all benchmarking problems into two groups based on the size of the extracted model, i.e., the number of extracted production rules or leaf nodes. In case of the car and nurse datasets the proposed classification method extracts much smaller models when compared with models generated by RIPPER, C4.5, and PART; the other considered biologically inspired algorithms also generate similarly small models. While the model is more compact its predictive accuracy is also lower. To indicate this difference we summarize the results for all benchmarking problems and the set of benchmarking problems that excludes the car and nurse datasets (indicated using * in Table 5, Table 6, and Table 8).

Table 5 and Table 6 report the mean results for both considered groups of the benchmarking problems. We average all measurements (the accuracy of the extracted model and the number of production rules or leaf nodes in the generated classification model) and the corresponding standard deviation values obtained from the individual datasets to provide the overview of the results.

The benchmarking problems are pre-processed to satisfy the requirement of the considered learners. We remove all unknown cases as the PSO II algorithm cannot use the cases with missing values to extract the classification model. We also use the same training and tests sets for each classifier (which is necessary to compute the T-test, which is explained later in this section) when performing the 10-fold cross validation.

We report the predictive accuracy of the classification models generated by the six considered learners over the eleven selected benchmarking problems in Table 5. We show the highest accuracy values for each individual dataset in bold and we provide the summarized mean results over all considered experiments.

Dataset	Algorithm					
	RIPPER	C4.5	PART	PSO II	AntMiner	Proposed method
bc	74.4 ± 8.46	75.9 ± 8.05	68.5 ± 7.66	74.8 ± 8.67	**77.6 ± 10.2**	70.8 ± 7.34
bcw	93.8 ± 2.67	93.8 ± 2.84	93.8 ± 2.84	93.4 ± 3.16	90.2 ± 2.06	**96.1 ± 2.06**
car	87.5 ± 1.86	92.3 ± 1.24	**95.4 ± 1.96**	78.8 ± 2.69	85.3 ± 0.35	80.4 ± 2.12
cmc	46.6 ± 2.65	49.2 ± 4.10	48.6 ± 3.59	43.2 ± 3.92	43.0 ± 3.80	**50.8 ± 4.02**
derm	89.4 ± 6.74	93.4 ± 4.05	**95.1 ± 3.82**	70.5 ± 4.63	95.0 ± 4.89	90.6 ± 2.85
dna	**93.9 ± 1.21**	93.4 ± 1.00	93.7 ± 1.24	83.8 ± 1.93	87.4 ± 1.24	90.6 ± 1.36
lymph	77.9 ± 10.9	**80.7 ± 8.94**	**80.7 ± 10.7**	78.0 ± 11.1	75.0 ± 9.88	80.1 ± 6.71
mush	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	96.8 ± 0.94	97.6 ± 0.87	99.4 ± 0.43
nurse	96.8 ± 0.54	97.1 ± 0.57	**99.1 ± 0.32**	77.3 ± 1.91	86.6 ± 0.72	79.8 ± 1.45
vote	**97.0 ± 4.12**	96.5 ± 3.99	95.2 ± 3.81	94.9 ± 3.72	96.1 ± 4.14	96.7 ± 4.12
zoo	88.0 ± 11.3	92.0 ± 7.89	93.0 ± 8.23	**94.2 ± 5.12**	91.8 ± 7.69	93.6 ± 4.97
MEAN	85.9 ± 4.59	87.7 ± 3.88	87.6 ± 4.01	80.5 ± 4.34	84.1 ± 4.51	84.5 ± 3.40
MEAN*	84.5 ± 5.34	86.1 ± 4.54	85.4 ± 4.62	81.2 ± 4.80	83.7 ± 5.04	85.4 ± 3.76

Table 5: Accuracy ± standard deviation results for the six learners on the eleven benchmarking problems. The results are based on the 10-fold cross-validation tests, which for PSO II, AntMiner, and the proposed method were repeated five times. The best results for each datasets are shown in bold. Mean values are reported for eleven or (*) nine benchmarking problems (excluding car and nurse datasets).

The proposed biologically inspired algorithm twice obtains the best accuracy (for the bcw and cmc datasets). The two competing biologically inspired algorithms, PSO II and AntMiner, obtain the best results for the zoo and bc datasets, respectively. PART is the learner that obtains the best accuracies for five out of the eleven datasets, which is the highest count of wins.

The proposed method obtains favourable accuracy for the datasets car and nurse that are excluded from the reduced group of nine benchmarking problems when compared to PSO II. Both algorithms use the same measure to evaluate the production rules during the rule extraction procedure and similar methods (BPSO and PSO, respectively) to extract single production rule. This improvement is likely due to the different rule representations.

The mean results show that the proposed biologically inspired classification algorithm obtains favourable accuracies when compared with PSO II and AntMiner. The accuracies are comparable when the proposed method is contrasted with RIPPER, C4.5 and PART for the reduced group of the nine benchmarking problems. The reported mean values may be misleading since different datasets are characterized by different default accuracies (accuracy when all cases are classified to the largest class, i.e., the class label with the highest number of cases in the training set). Therefore, later in this section we summarize the results using win/draw/loss counts and a statistical significance test.

We observe that the proposed classification algorithm has the smallest mean standard deviation. This shows that the accuracy of the models generated from the individual benchmarking problems is less variable for the proposed method than the accuracies of the other competing methods. This could be due to the generation of rules for each class in the case of the proposed classification algorithm.

We report the number of rules or leaf nodes in the classification models extracted by the six considered learners over the eleven selected benchmarking problems in Table 6. We show the smallest classification models for each individual dataset in bold and we provide the summarized mean results over all considered experiments.

Dataset	Algorithm					
	RIPPER	**C4.5**	**PART**	**PSO II**	**AntMiner**	**Proposed method**
bc	**3.60 ± 0.96**	12.1 ± 10.9	18.4 ± 4.20	6.10 ± 0.65	6.26 ± 0.46	6.00 ± 0.00
bcw	13.0 ± 1.41	49.6 ± 7.59	11.5 ± 2.68	9.50 ± 1.08	12.6 ± 0.52	**2.00 ± 0.42**
car	35.5 ± 5.73	122. ± 6.47	62.0 ± 3.05	8.12 ± 0.31	18.0 ± 0.75	**6.20 ± 0.42**
cmc	**6.40 ± 1.90**	182. ± 76.2	187. ± 11.1	7.12 ± 0.60	11.9 ± 0.91	9.00 ± 0.00
derm	14.0 ± 1.56	23.9 ± 4.40	8.10 ± 1.66	9.64 ± 0.70	7.82 ± 0.59	**6.12 ± 0.14**
dna	17.8 ± 2.82	78.9 ± 4.15	65.3 ± 3.89	6.74 ± 0.38	8.68 ± 0.50	**5.22 ± 0.24**
lymph	6.90 ± 1.20	18.4 ± 1.40	9.40 ± 1.26	**6.12 ± 0.45**	6.54 ± 0.42	6.22 ± 0.41
mush	6.00 ± 0.00	24.6 ± 0.52	11.3 ± 0.67	4.98 ± 0.22	7.88 ± 0.43	**2.80 ± 0.28**
nurse	112. ± 9.43	359. ± 8.19	189. ± 11.2	7.02 ± 0.06	17.7 ± 0.69	**7.00 ± 0.00**
vote	**2.00 ± 0.00**	2.50 ± 1.08	5.30 ± 0.95	3.66 ± 0.79	4.32 ± 0.42	2.60 ± 0.52
zoo	7.30 ± 0.67	10.7 ± 2.79	7.60 ± 0.52	7.12 ± 0.19	**6.00 ± 0.00**	7.00 ± 0.00
MEAN	20.5 ± 2.34	80.5 ± 11.24	52.3 ± 3.74	6.92 ± 0.49	9.78 ± 0.52	5.49 ± 0.22
MEAN*	8.56 ± 1.17	44.8 ± 12.11	36.0 ± 2.99	6.76 ± 0.56	8.00 ± 0.47	5.24 ± 0.19

Table 6: Number of rules or leaf nodes ± standard deviation in models generated by six learners on the eleven benchmarking problems. The results are based on the 10-fold cross-validation tests, which for PSO II, AntMiner, and the proposed method were repeated five times. The best results, smallest number of rules (or leaf nodes) on the extracted model for each datasets are shown in bold. Mean values are reported for eleven or (*) nine benchmarking problems (excluding car and nurse datasets).

The proposed biologically inspired algorithm extracts the smallest classification models, in terms of the number of production rules or leaf nodes, for six out of the eleven datasets (including the car and nurse datasets that are excluded from the reduced group of the benchmarking problems), which is the highest count of wins. The competing biologically inspired algorithms, PSO II and AntMiner, obtain the best result for the lymph and zoo datasets, respectively. RIPPER is the learner that extracts the smallest classification models for the bc, cmc and vote datasets.

The AntMiner algorithm, in the case of the zoo dataset, extracts the classification model that contains only six production rules, including the default rule. While the dataset contains seven unique class labels, the extracted model does not discriminate the test cases between all the classes that are present in the training set. Later in this section we provide a more detailed analysis of the completeness of the classification model.

The mean results show that the proposed biologically inspired classification algorithm obtains favourable results in terms of the size of the extracted classification model when contrasted with the considered classification algorithms. The mean value for all selected benchmarking problems equals 5.49 and is close to the average number of classes in the eleven selected datasets that equals 3.64 . This is likely due to the proposed rule representation and the generalization properties of the fitness function that is used during the optimization procedure to obtain a single production rule. While the classification model extracted by the proposed method contains fewer prediction rules, the rule representation is more complex. Later in this section we summarize the above results using win/draw/loss counts.

We observe that the proposed classification algorithm has the smallest mean standard deviation. This shows that the size of the models extracted from the individual benchmarking problems is less variable in the proposed method than the size of the extracted models in case of other competing methods. This could be due to generation of rules for each class in the case of the proposed classification algorithm, which is in contrast to the other considered methods.

We report the summarized win/draw/loss results for the six considered learners over the eleven selected benchmarking problems in Table 7. Table summarizes the results reported in Table 5 and Table 6. We provide the comparison of the accuracy of the extracted models on the upper triangle and the size of the generated classification model in the lower triangle.

Algorithm	Algorithm					
	RIPPER	**C4.5**	**PART**	**PSO II**	**AntMiner**	**Proposed method**
RIPPER	-	2/2/7 (2/2/5)	3/2/6 (3/2/4)	8/0/3 (6/0/3)	8/0/3 (6/0/3)	6/0/5 (4/0/5)
C4.5	11/0/0 (9/0/0)	-	3/3/5 (3/3/3)	10/0/1 (8/0/1)	9/0/2 (7/0/2)	7/0/4 (5/0/4)
PART	9/0/2 (7/0/2)	3/0/8 (3/0/6)	-	9/0/2 (7/0/2)	9/0/2 (7/0/2)	6/0/5 (4/0/5)
PSO II	3/0/8 (3/0/6)	1/0/10 (1/0/8)	1/0/10 (1/0/8)	-	4/0/7 (4/0/5)	2/0/9 (2/0/7)
AntMiner	4/0/7 (4/0/5)	1/0/10 (1/0/8)	1/0/10 (1/0/8)	9/0/2 (7/0/2)	-	4/0/7 (2/0/7)
Proposed method	3/0/8 (3/0/6)	1/0/10 (1/0/8)	0/0/11 (0/0/9)	2/0/9 (2/0/7)	1/0/10 (1/0/8)	-

Table 7: Win/draw/loss results for six learners on the eleven benchmarking problems. The results in the brackets are based on the nine benchmarking problems (excluding car and nurse datasets). The upper triangle reports the accuracy of the generated classification model and the lower triangle reports the number of rules (or leaf nodes) in the extracted classification models, e.g., the 3/0/8 value located in row PART and column C4.5 means that PART model contains more rules (or leaf nodes) than C4.5 model on 3 datasets and PART model contains less rules (or leaf nodes) than C4.5 model on 8 datasets, the 3/3/5 value in the C4.5 row and PART column mean that C4.5 accuracy is higher than PART accuracy three times, the accuracy of the two methods is even three times, and C4.5 accuracy is lower than PART accuracy five times. The results are based on the 10-fold cross-validation tests, which for PSO II, AntMiner, and the proposed method were repeated five times.

Table 7 provides an overview of the experimental comparison between all selected algorithms. However the most relevant part of the summary is the last row and the last column that compare the proposed biologically inspired classification algorithm with other considered classification algorithms such as RIPPER, C4.5, PART, PSO II, and AntMiner.

The proposed method obtains favourable accuracy when compared with PSO II and AntMiner, i.e., the extracted classification model has better accuracy in case of nine out of eleven and seven out of eleven benchmarking problems, respectively. The win/draw/loss accuracy measure of the proposed method is comparable with the RIPPER and PART when considering both complete set of eleven benchmarking problems and the reduced set of nine benchmarking problems. C4.5 obtains better win/loss accuracy record against the proposed method in the case of both sets of benchmarking problems.

We further summarize Table 7 by computing a sum of wins between a given learner and all other classification algorithms, including both the complete set of eleven benchmarking problems and the reduced set of nine benchmarking problems (results reported in brackets). The results are: 36(28), 35(25), 30(28), 26(21), 19(16), and 12(8) for C4.5, PART, the proposed method, RIPPER, AntMiner, and PSO II classification algorithms, respectively. While the proposed method is placed third when using the complete set of eleven datasets, it scores first together with the C4.5 algorithm when considering the reduced set of nine datasets.

We observe that models extracted by the proposed method contains fewer production rules than the classification models extracted by the other classification algorithms for a larger number of dataset. This is consistent over all considered competing methods, which can be observed in the last row in Table 7. However, the other learners have simpler representation of the production rules.

We report results of the T-test comparison in Table 8. We test whether the differences in the quality between models extracted by our method and each competing learner are statistically significant. Each of the compared models is generated from the same data (the same fold). This means that we perform the 10-fold cross validation with exactly the same division of the benchmarking problems for each considered learner.

Dataset	Algorithm				
	RIPPER	C4.5	PART	PSO II	AntMiner
bc	~~	~~	~~	~~	~~
bcw	++	++	++	++	++
car	--	--	--	~~	--
cmc	++	~~	~~	++	++
derm	~~	~~	--	++	--
dna	--	--	--	++	++
lymph	~~	~~	~~	~~	~~
mush	--	--	--	++	++
nurse	--	--	--	++	--
vote	~~	~~	~~	~~	~~
zoo	~~	~~	~~	~~	~~
SUMMARY	2/5/4	1/6/4	1/5/5	6/5/0	4/4/3
SUMMARY*	2/5/2	1/6/2	1/5/3	5/4/0	4/4/1

Table 8: T-test results of accuracy comparison between the proposed classification algorithm and five competing learners on the eleven benchmarking problems; "++" indicates that the proposed method is significantly better than the competing learner, "~~" indicates no significant difference, "--" indicates that the proposed method is significantly worse than the competing learner. Summary values reported as "++/~~/--" respectively summarize the results for eleven or (*) nine benchmarking problems (excluding car and nurse datasets), e.g., the 4/4/1 value located in the row SUMMARY* and column AntMiner means that the proposed method is significantly better than AntMiner on 4 datasets, there is no significant difference between the proposed method and AntMiner on 4 datasets, and AntMiner is significantly better than the proposed method once. The results are based on the 10-fold cross-validation tests, which for PSO II, AntMiner, and the proposed method were repeated five times.

The proposed biologically inspired classification algorithm obtains favourable results in terms of statistical significance when compared with the other two considered biologically inspired algorithms. However, the results favour the competing traditional "white-box" learners when we consider the complete set of eleven benchmarking problems. The accuracy comparison shows that RIPPER and C4.5 algorithms are comparable to the proposed method when considering the reduced set of nine benchmarking problems.

Table 9 contains the summary of completeness analysis of the classification models extracted from the benchmarking problems. We analyze whether the classification models contain a complete set of rules, i.e., the generated ruleset contains non-default production rules for all class labels present in the training set.

During rule extraction procedure, RIPPER always selects the majority class in the training set and creates a default rule for the class. It means that for each benchmarking problem in Table 9, the column with RIPPER algorithm contains at least 10 folds with one class label that has only the default production rule.

Dataset	Algorithm					
	RIPPER	C4.5	PART	PSOII	AntMiner	Proposed method
bc	10/0	0/0	0/0	0/0	0/0	0/0
bcw	10/0	0/0	0/0	0/0	0/0	0/0
car	10/0/0/0	0/0/0/0	0/0/0/0	0/0/0/9	0/0/0/0	0/0/0/0
cmc	10/0/0	0/0/0	0/0/0	0/0/2.2	0/0/0	0/0/0/0
derm	10/0/0/0/0/0	0/0/0/0/0/0	0/0/0/0/0/5	0/0/0/0/0.2/1	0/0/0/0/0/0	0/0/0/0/0/0
dna	0/0/10	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0
lymph	2/10/0/1	0/0/0/1	6/0/0/3	10/0/0/3	3/0/0/2.2	0/0/0/0
mush	10/0	0/0	0/0	0/0	0/0	0/0
nurse	9/4/0/1/0	0/10/0/0/0	0/8/0/0/0	0/10/9.8/0/0	0/5.6/0/0/0	0/0/0/0/0
vote	10/0	0/0	0/0	0/2.2	0.2/0	0/0
zoo	10/0/0/0/2/0/0	0/0/0/0/0/0/0	0/0/2/0/3/0/0	0/0/1/0/8/0/0	0/0/10/0/10/0/0	0/0/0/0/0/0/0

Table 9: Completeness of models generated by the learners on the benchmarking problems. The reported values correspond to the number of folds with missing or only default rule per class label, e.g. the 2/10/0/1 value located in row lymph and column RIPPER means that 2 folds for class 0, 10 folds for class 1, 0 folds for class 2, and 1 fold for class 3 include only the default or no rules. The results are based on the 10-fold cross-validation tests, which for PSO II, AntMiner, and the proposed method were repeated five times. Because of that the results for PSO II, AntMiner, and the proposed method are averaged and may be a floating point numbers.

The only method that contains non-default production rules for all class labels present in the training set is the proposed biologically inspired classification algorithm. This is a result of the design of the proposed rule extraction procedure. The other competing learners extract the classification models that are incomplete for at least two benchmarking problems.

The classification models extracted on the lymph, nurse, and zoo benchmarking problems are the most problematic in terms of the model completeness for the RIPPER, C4.5, PART, PSO II, and AntMiner algorithms. The three datasets are a multi-class problems with the uneven distribution of cases among the class labels. The classification models do not contain the non-default production rules for the classes with the smallest number of cases.

The proposed biologically inspired algorithm is designed to provide the complete set of production rules. The eleven selected real world benchmarking problems show that completeness of the model is not guaranteed in the case of the considered five competing classifiers.

5.6 Summary

In this chapter we experimentally compared the proposed biologically inspired classification method with the existing “white-box” classification methods, such as RIPPER, C4.5, PART, and the current biologically inspired classification algorithms such as PSO II and AntMiner. First, we parametrized the proposed method to obtain a set of BPSO parameters for the subsequent experiments. Next, we used two synthetic datasets to provide an extensive comparison of the three considered biologically inspired algorithms and discuss the extracted classification models. Finally, we performed the experimental comparison of the six considered classifiers on the eleven selected benchmarking problems. We analyzed the performance of the methods in terms of the predictive accuracy, the number of the extracted production rules or leaf nodes, and the completeness of the extracted classification model.

The comparisons showed that the proposed method is comparable in terms of predictive accuracy to the traditional “white-box” classifiers and favourable when contrasted with the two considered biologically inspired algorithms. Additionally, we experientially confirmed that the proposed method extracts a complete set of modular production rules, which is in contrast to the other five competing classifiers.

6 Conclusions

In this work we discussed the use of biologically inspired algorithms to generate "white-box" classification models. Biologically inspired algorithms are heuristic-based optimization methods that provide global search strategy and use population of individuals to find approximate solution of a given problem. They provide an interesting alternative for generation of classification rules when compared with traditional greedy search-based approaches.

We described the differences between the biologically inspired classification algorithms such as PSO II and AntMiner, including the rule representation, the encodings scheme, the rule extraction procedure, and the classification procedure.

In this work we proposed enhancements with respect to the rule representation, the rule extraction procedure, and the classification procedure of the current algorithms and we introduce novel biologically inspired classification algorithm. We considered several the properties of generated classification models, which is in contrast to the existing methods that aim at obtaining the highest possible classification accuracy. Instead of focusing only on the accuracy, we analyzed three other properties of the classification models such as the total number of rules, modularization, and the completeness of the extracted model.

Extensive experimental tests showed that the proposed method provides favourable predictive accuracy when compared with the current biologically inspired classification algorithms and comparable predictive accuracy when contrasted with three traditional "white-box" algorithms such as RIPPER, C4.5, and RIPPER. Additionally, the proposed method provides a complete set of modular production rules, which is in contrast to the current biologically inspired classification algorithms.

List of significant contributions:

- Development of a novel classification algorithm to generate production rules using Binary Particle Swarm Optimization algorithm.
- Novel rule encoding method in context of current biologically inspired classification algorithms.
- Development of biologically inspired classification algorithm that generates a set of modular rules.
- Development of biologically inspired classification algorithm that generates rules for each class present in the training set.
- The parametrization of the Binary Particle Swarm Optimization algorithm.
- The first comprehensive comparison of several biologically inspired classifiers.
- The first comparative study that includes rule-based, decision-tree based, decision-list based, and the biologically inspired classification methods.
- The first study to consider and compare the completeness of the generated classification models.

7 Future work

In this work we discussed the design of the biologically inspired classification algorithm that extracts a complete set of modular production rules and provided an experimental comparison with other "white-box" classification algorithms. During the development of the proposed method we found a number of interesting directions that can be analyzed in future:

- Analysis of the complexity of the extracted production rules. In this work we proposed extensions to the rule representations used in the current biologically inspired algorithms, but we did not analyze the complexity of the production rules as focused on the modularity of the generated rules.
- Processing of continuous attributes. It would be interesting to extend the proposed algorithm to cope with the continuous attributes, as currently the proposed method can work only with the discrete attributes.
- Scalability analysis. In this work we only briefly mentioned the computational costs of the proposed method, measured as the running time. It would be interesting to perform an extensive analysis and comparison between the biologically inspired algorithms in terms of the computational costs.

8 References

[1] M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From data mining to knowledge discovery: an overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*. AAAI/MIT, 1996.

[2] L.A. Kurgan and P. Musilek. A survey of Knowledge Discovery and Data Mining process models. *Knowledge Engineering Review*, 21(1):1-24, 2006.

[3] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*, 2005. Springer-Verlag New York, Inc., 2005.

[4] K.J. Cios, W. Pedrycz, R.W. Swiniarski and L.A. Kurgan. *Data Mining: A Knowledge Discovery Approach*, 2007. Springer-Verlag New York, Inc., 2007.

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*, 1973. John Wiley and Sons, 1973.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273-297, 1995.

[7] W.W. Cohen. Fast Effective Rule Induction. *Proceedings of the 12th International Conference on Machine Learning*, pages 115-123. 1995.

[8] J.R. Quinlan. *C4.5: programs for machine learning*, 1993. Morgan Kaufmann Publishers Inc., 1993.

[9] F. Eibe and I.H. Witten. Generating accurate rule sets without global optimization. *Proceedings of the Fifteenth International Conference on Machine Learnin*, pages 144-151. 1998.

[10] J. Sietsma and R.J.F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67-79, 1991.

[11] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris and D.M. Hummels. On the

training of radial basis function classifiers. *Neural Networks*, 5(4):595-603, 1992.

[12] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989. Addison-Wesley Longman Publishing Co., Inc., 1989.

[13] J. Kennedy and R. Eberhart. Particle Swarm Optimization. *IEEE International Conference on Neural Networks - Conference Proceedings*, pages 1942-1948. 1995.

[14] Shi Yuhui and R. Eberhart. A Modified Particle Swarm Optimizater. *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 69-73. 1998.

[15] M. Dorigo, V. Maniezzo and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29-41, 1996.

[16] A.A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. In A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computation*. Springer-Verlag, 2002.

[17] Tiago Sousa, Arlindo Silva and Ana Neves. Particle swarm based Data Mining Algorithms for classification tasks. *Parallel Computing*, 30(5-6):767-783, 2004.

[18] R.S. Parpinelli, H.S. Lopes and A.A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321-332, 2002.

[19] L.A. Breslow and D.W. Aha. Simplifying decision trees: A survey. *Knowledge Engineering Review*, 12(1):1-40, 1997.

[20] H. Aytug, M. Khouja and F.E. Vergara. Use of genetic algorithms to solve production and operations management problems: A review. *International Journal of Production Research*, 41(17):3955-4009, 2003.

[21] R. Leardi. Genetic algorithms in chemometrics and chemistry: a review. *Journal of Chemometrics*, 15(7):559 - 569, 2001.

[22] J. Kennedy and R. Eberhart. *Swarm intelligence*, 2001. Morgan Kaufmann Publishers Inc., 2001.

[23] J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 4104-4108. 1997.

[24] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53-66, 1997.

[25] M. Dorigo and G. Di Caro. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, 1999.

[26] M. Dorigo. Ant Algorithms Solve Difficult Optimization Problems. *Proceedings of the 6th European Conference on Advances in Artificial Life*, pages 11-22. 2001.

[27] T. Stutzle and H.H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8):889-914, 2000.

[28] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58-73, 2002.

[29] R. Eberhart and Y. Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 84-88. 2000.

[30] T.M. Cover and J.A. Thomas. *Elements of information theory*, 1991. Wiley-Interscience, 1991.

[31] H.S. Lopes, M.S. Coutinho and W.C. Lima. An evolutionary approach to simulate cognitive feedback learning in medical domain. In E. Sanchez, T. Shibata and L.A. Zadeh, editors, *Genetic Algorithms and Fuzzy Logic Systems*.

[32] M.V. Fidelis, H.S. Lopes and A.A. Freitas. Discovering Comprehensible Classification Rules with a Genetic Algorithm. *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 805-810. 2000.

[33] W.M. Spears and D.F. Gordon. Is consistency harmful? 1992.

[34] K.A. De Jong, W.M. Spears and D.F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161-188, 1993.

[35] Newman, D.J., S. Hettich, C.L. Blake and C.J. Merz. **UCI** Repository of machine learning databases. 1998.

[36] I.H. Witten and F. Eibe. *Data Mining: Practical machine learning tools and techniques*, 2005. Morgan Kaufmann, 2005.