A Fuzzy Discrete Particle Swarm Optimization Classifier for Rule Classification

Min Chen and Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, ND, USA
min.chen@my.ndsu.edu, simone.ludwig@ndsu.edu

***Abstract*—The need to deduce interesting and valuable information from large, complex, information-rich data sets is common to many research fields. Rule discovery or rule mining uses a set of IF-THEN rules to classify a class or category in a comprehensible way. Besides the classical approaches, many rule mining approaches use biologically-inspired algorithms such as evolutionary algorithms and swarm intelligence approaches. In this paper, a Particle Swarm Optimization based discrete classification implementation with a local search strategy (DPSO-LS) was devised and applied to discrete data. In addition, a fuzzy DPSO-LS (FDPSO-LS) classifier is proposed for both discrete and continuous data in order to manage imprecision and uncertainty. Experimental results reveal that DPSO-LS and FDPSO-LS outperform other classification methods in most cases based on rule size, True Positive Rate (TPR), False Positive Rate (FPR), and precision, showing slightly improved results for FDPSO-LS.**

***Keywords*-Fuzzy rule-based classification system, Pittsburgh approach, particle swarm optimization, local strategy**

I. Introduction

In this current information age, a tremendous expansion in the volume of data is seen that is being generated and stored. The need to understand large, complex, information-rich data sets is common to all fields of studies. Given this tremendous amount of data, efficient and effective tools need to be available to analyze and reveal valuable knowledge that is hidden. The objective of the field of knowledge discovery and data mining is the discovery of knowledge that is not only correct, but also comprehensible.

The two primary goals of data mining can be classified as *prediction* and *description* [1]. *Prediction* involves using some features or fields of the data set to predict unknown or future values of interest, whereas *description* focuses on finding patterns describing the data that can be interpreted by humans. Several data mining techniques using prediction and description have emerged that include classification, clustering, regression, dependence modeling, etc. The classification technique is used to discover a predictive learning function that classifies a data item into several predefined classes. It is also known as supervised classification, whereby given class labels are ordered to objects in the data collection. In general, classification approaches use a training set in which all objects are already associated with their corresponding class labels. The classification algorithm then learns from the training set data and builds a model. This model is then used to classify unseen data and to assign a class label to each data item.

Rule discovery is an important classification method that has been attracting a significant amount of researchers in recent years. It uses a set of IF-THEN rules to classify a class or category in a natural way. A rule consists of antecedents and a consequent. The antecedents of the rule consist of a set of attribute values and the consequent of the rule is the class which is predicted by that rule.

Frequently used classification methods include decision trees, neural network, and naive Bayes classification, etc. [2]. Decision trees specify the sequences of decisions that need to be made accompanied by the resulting recommendation. Typically, a decision-tree learning approach uses a top-down strategy. Information gain was introduced as a "goodness" criterion first in a decision tree algorithm known as ID3 [3]. However, since then, ID3 has been further improved and is now known as C4.5 [4]. These improvements include methods for dealing with numeric attributes, missing values, and noisy data. Neural networks is another method frequently used in data mining. It is a black box system with layers of neurons that learn the task by applying input and output values. Neural networks are seen as data driven self-adaptive methods. They can adjust themselves to the data without any explicit specification of the underlying model [5]. Naive Bayes learning is one particular strategy belonging to the category of neural networks. It is a statistical method for classification, which is based on applying the Bayes theorem with naive independence assumption [6].

Fuzzy logic can improve the classification system by using fuzzy sets to define overlapping class definitions [7]. The interpretability of the results can be improved and more insight into the classifier structure and decision making process is provided by the application of fuzzy IF-THEN rules [8]. Fuzzy rules are linguistic IF-THEN constructs that have the general form "IF A THEN C", where A and C are collections of propositions and postpositions containing linguistic variables. A is called the antecedent, and C is the consequent of the rule. In effect, the tolerance for imprecision and uncertainty is exploited through granulation in soft data compression by using linguistic variables and fuzzy IF-THEN rules [8]. In this respect, fuzzy logic has the feature of mimicking the essential ability of the human mind to summarize data and focus on decision-relevant information.

In a more explicit form, the i^{th} rule has the following form:

$$\text{IF } x_{i1} \in A_1^m \text{ AND .. AND } x_{ij} \in A_j^n \text{ THEN } c_i \in C_i^k \quad (1)$$

where x_{ij} denotes the j^{th} attribute of the i^{th} rule. A_j^m denotes the m^{th} antecedent value of the j^{th} attribute. c_i is the consequent of the i^{th} rule.

Due to its simplicity and capability, Particle Swarm Optimization (PSO) is a biology-inspired algorithm which has been widely applied in different areas. This paper proposes a discrete particle swarm optimization with a local strategy (DPSO-LS) for solving the classification problem. The local search strategy helps to overcome local optima in order to improve the solution quality. The DPSO-LS uses the Pittsburgh approach whereby a rule base is used to represent a 'particle'. Furthermore, since the DPSO-LS can only be applied on discrete data, an additional classifier called Fuzzy DPSO-LS (FDPSO-LS) classifier is implemented on both discrete and continuous data to tolerate imprecision and uncertainty. This paper is an extension of [9] whereby the FDPSO-LS algorithm is proposed and evaluated.

The remainder of the paper is arranged as follows. Section II describes related work. The proposed two approaches DPSO-LS and FDPSO-LS are introduced in Section III. The experimental setup and results of the two approaches are demonstrated in Section IV. Finally, conclusions and future work are discussed in Section V.

II. Related Work

Related work in classification rule mining using biology-inspired algorithms mainly include evolutionary algorithms and swarm intelligence approaches. Genetic algorithm (GA) based concept learning uses either the Pittsburgh approach or the Michigan approach [10]. For the Pittsburgh approach, every individual in the GA is a set of rules that represents a complete solution to the learning problem. For the Michigan approach, each individual represents a single rule that provides only a partial solution to the overall learning task.

GA-based concept learning has been widely used for rule mining. In [10], a GA-based algorithm is proposed to discover comprehensive IF-THEN rules. It uses a flexible chromosome encoding where each chromosome corresponds to a classification rule. In addition, a hybrid decision tree/-genetic algorithm is used to discover small disjunct rules in [11]. A decision-tree algorithm is used to classify examples belonging to large disjuncts, while a new GA is designed for classifying examples belonging to small disjuncts.

Evolutionary approaches for automated discovery of censored production rules, augmented production rules and comprehensible decision rules are introduced in [12], [13], [14], respectively. The proposed GA-based approaches, similarly, use a flexible chromosome encoding, where each chromosome corresponds to an augmented production rule, a comprehensible decision rule or a censored production rule. An Evolutionary Multiobjective Optimization (EMO) algorithm is used to search for a large number of non-dominated fuzzy rule-based classifiers in [15].

With regards to swarm intelligence approaches, a classification algorithm called Ant-Miner, first introduced in [16], has been successfully applied to rule classification problems. PSO is another approach inspired by nature. However, most of the swarm intelligence algorithms for rule classification are based on the Michigan approach ([17], [18]).

Related work in fuzzy classification rule mining using the biology-inspired algorithms mainly include evolutionary algorithms and swarm intelligence approaches. GA is a popular evolutionary algorithm, which has been employed for the learning of fuzzy rules. GAs have been applied to learn both antecedent and consequent of fixed or varying number of fuzzy rules [19], [20], [21]. Also, GAs have been combined with other techniques like neural networks [22], Kalman filters [23], hill climbing [24], and fuzzy clustering [22]. EMO algorithms, which generate a family of equally valid solutions, have been introduced in [25].

Ant Colony Optimization (ACO), one of the swarm intelligence techniques, has been successfully used to extract rule based classification systems. In [26], ACO is used to extract fuzzy IF-THEN rules for the diagnosis of diabetes. A combination of ACO and fuzzy set theory, named FACO-Miner, is applied to learn a set of fuzzy rules from labeled data in a parallel manner in [27]. An improved ACO technique using fuzzy inference rules is applied to image classification and analysis in [28].

With respect to PSO, a Pittsburgh-based PSO fuzzy system for knowledge acquisition is introduced in [29]. A modified PSO, called Mutation PSO (MPSO), is built and used to obtain an optimal fuzzy rule-base. The algorithm generated a compact fuzzy rule base that works efficiently for medical diagnosis problems [30]. In [31], a case study of intrusion detection using a PSO approach for evolutionary fuzzy rule learning is proposed. It is capable of detecting known intrusive behavior in a computer network with an acceptable performance.

PSO has been proven to be able to achieve a faster convergence than the GA algorithm [29]. It has been experimentally shown that the PSO algorithm scales well and is not highly sensitive to the population size [29]. As far as the authors' knowledge is concerned, due to the lack of flexibility of the Pittsburgh approach [32], the Pittsburgh-based PSO algorithm on rule classification is rarely used in literature. On the other hand, in order to avoid premature convergence of particles, the Michigan approach usually requires some changes in the definition of the PSO algorithm to repel a particle from its neighbor [32]. In addition, the Michigan approach aims to optimize each rule's quality individually, and does not take the interaction between other rules into account [17]. In [29], the knowledge acquisition with a Pittsburgh-based swarm-intelligence approach is introduced. A learning strategy of a fuzzy-rule-based meta-scheduler is analyzed and compared with other scheduling strategies. In our study, similarly, we propose a Pittsburgh-based swarm-intelligence method, however, we improve the classification by applying a local strategy to address PSO's convergences problem. Furthermore, in order for the method to handle imprecision and vagueness in data sets fuzzy logic is employed.

III. Proposed Approaches

Two classifiers are proposed and investigated: a DPSO-LS classifier and a fuzzy DPSO-LS classifier (abbreviated as FDPSO-LS). The DPSO-LS classifier is designed to classify discrete data sets. As we have mentioned above, the use of linguistic variables and fuzzy IF-THEN rules exploits the tolerance for imprecision and uncertainty. In this respect, we extend the DPSO-LS classifier to a fuzzy DPSO-LS (FDPSO-LS) classifier, which can classify both discrete and continuous data sets. In this section, we first describe the DPSO-LS algorithm followed by a detailed description of the FDPSO-LS classifier.

A. Discrete Particle Swarm Optimization with Local Strategy (DPSO-LS)

PSO was introduced by Eberhart and Kennedy [33] and is based on the analogy of the behavior of flocks of birds or schools of fish. Although the PSO algorithm was proposed for continuous space problems, however, many real-world datasets use categorical data, and therefore, we considered this within our classification task formulation. In classical PSO, swarm individuals are called particles, and the population is called the swarm. Each particle has its own position, velocity and historical information. The particles fly through the search space by using their own as well as their neighbors' historical information to steer toward the local or global optima.

In particular, a discrete PSO approach (DPSO-LS) for the classification rule mining problem is proposed. A Rule Base (RB) as a whole represents a 'particle'. Each RB is denoted as a matrix, where each row describes a classification rule. The rules are IF-THEN rules consisting of conjunctive antecedents and one consequent. Hence, the i^{th} particle is presented as follows:

$$P_i = \begin{pmatrix} a^i_{1,1} & a^i_{1,2} & ... & a^i_{1,n} & c^i_1 \\ a^i_{2,1} & a^i_{2,2} & ... & a^i_{2,n} & c^i_2 \\ ... & ... & ... & ... & ... \\ a^i_{m,1} & a^i_{m,2} & ... & a^i_{m,n} & c^i_m \end{pmatrix} \tag{2}$$

where a^i_{mn} represents the n^{th} antecedent in the m^{th} rule of the i^{th} particle. c^i_m is the m^{th} consequent of the i^{th} particle. m is the number of rules, and n is the number of antecedents. Thus, a particle consists of m rules, where each rule has n antecedents and 1 consequent.

The values of every antecedent are enumerated consecutively starting from 1. In this work, an antecedent has 3 discrete values, it will be enumerated as $\{1, 2, 3\}$. In this way, 0 means the antecedent is ignored. Thus, a rule with all its antecedents having a value of 0 is not allowed. In addition, the constraints of the swarm position updating process need to be considered since the particle might fly outside the solution space:

$$a^i_{j,k} \in [0, NF_{in}], j \in \{1, 2, ..., m\} \tag{3}$$

$$k \in \{1, 2, ..., n\} \tag{4}$$

$$c^i_j \in [1, NF_{out}] \tag{5}$$

where NF_{in} and NF_{out} represent the number of discrete values for an antecedent and a consequent, respectively. The i^{th} particle's velocity matrix is denoted as follows:

$$V_i = \begin{pmatrix} v^i_{1,1} & v^i_{1,2} & \cdots & v^i_{1,n} & v^i_{1,n+1} \\ v^i_{2,1} & v^i_{2,2} & \cdots & v^i_{2,n} & v^i_{2,n+1} \\ ... & ... & ... & ... & ... \\ v^i_{m,1} & v^i_{m,2} & \cdots & v^i_{m,n} & v^i_{m,n+1} \end{pmatrix} \tag{6}$$

where $v^i_{j,k} \in [V_{min}, V_{max}]$, $j \in \{1, 2, ..., m\}$, and the velocity matrix has the same dimension as the position matrix. V_{min} and V_{max} are the minimum and maximum values allowed for the velocity, respectively. More specifically, we use a change vector $\vec{V_i}$, which is the change vector for the i^{th} particle with the same dimension as the velocity matrix.

$$\vec{V_i} = \begin{pmatrix} \hat{v}^i_{1,1} & \hat{v}^i_{1,2} & \cdots & \hat{v}^i_{1,n} & \hat{v}^i_{1,n+1} \\ \hat{v}^i_{2,1} & v^i_{2,2} & \cdots & v^i_{2,n} & v^i_{2,n+1} \\ ... & ... & ... & ... & ... \\ \hat{v}^i_{m,1} & \hat{v}^i_{m,2} & \cdots & \hat{v}^i_{m,n} & \hat{v}^i_{m,n+1} \end{pmatrix} \tag{7}$$

The values of $\vec{V_i}$ are randomly assigned to 1, 2 and 3, where 1, 2 and 3 are denoted as three directions. 1 is denoted as the direction of the particle's movement from the current position to the local best position ($Pbest$). 2 is denoted as the direction of the particle's movement from the current position to the global best position ($Gbest$). 3 is denoted as the direction of the particle's movement from the current position to another position at random within a specified range. The three directions are randomly assigned by following the ratios ω_1, ω_2, and ω_3 ($\omega_1 < \omega_2 < \omega_3$). As shown in Equation 8, the sum of the ratios should be equal to one. By adopting the concept of change vector, the velocity of the particle can be updated by considering the local best position, global best position and random changes. Precisely, as shown in Equation 9, for the i^{th} particle, $V_1(t)$ is the difference between the local best position and the current position, while $\vec{V_i}$ consist of 1s, and the rest of the values in the matrix are set to 0. Similarly, $V_2(t)$ is the difference between the global best position and the current position, while $\vec{V_i}$ consist of 2s. Values of $V_3(t)$ are randomly assigned within a specified range (see Equation 10), while values of $\vec{V_i}$ consist of 3s at the same positions. $\oplus$ denotes a matrix addition.

$$\omega_1 + \omega_2 + \omega_3 = 1 \tag{8}$$

$$V(t+1) = V_1(t) \oplus V_2(t) \oplus V_3(t) \tag{9}$$

$$V_3(t) \in [V_{min}, V_{max}] \tag{10}$$

After the velocity has been calculated, the particle's position can be computed as follows:

$$P(t+1) = P(t) \oplus V(t+1) \tag{11}$$

1) Definition of Overall Fitness: We propose a rule selection method where the number of classification rules included in each rule set is fixed to a predefined number. That is, each rule set with a specific number of rules (a rule base) is a particle. Thus, the overall fitness function of the rule set can be defined as follows:

$$F(S) = Coverage = \frac{NCP(S)}{|S|} \tag{12}$$

where $NCP(S)$ is the number of instances that have been correctly classified in the data set S, and $|S|$ is the number of instances in the data set S.

2) Local Mutation Strategy: Since PSO, in general, can easily get stuck in local optima, a local strategy need to be devised that is run after a certain number of iterations has elapsed. In particular, the local strategy that was devised for DPSO-LS makes use of mutation. The proposed local strategy refines the worst rule of the best rule base, i.e., the global best position, in order to improve the overall performance every 20 iterations. Thus, for each selected worst rule, we mutate one value of the antecedent randomly within the constraints to see whether it improves the overall performance or not. If it improves the performance, we stop and replace the worst rule with the new rule. Otherwise, we continue mutating randomly until we have found a new rule or until we have mutated a maximum of 10 times.

The equation to measure the quality of every rule uses the Laplace-corrected precision [17] equation, which is given as:

$$f = \frac{1+TP}{1+TP+FP} \tag{13}$$

where TP is the number of True Positives, and FP is the number of False Positives. The equation is also used to prune the rules for which the f value is less than 0.1.

B. DPSO-LS Classifier

The proposed algorithm includes four main phases: data preprocessing phase, training phase, DPSO phase and testing phase. As shown in Fig. 1, the DPSO-LS classifier includes all the solid rectangles and excludes the red dashed rectangles (these are only used for FDPSO-LS). The four phases are described respectively as follows.

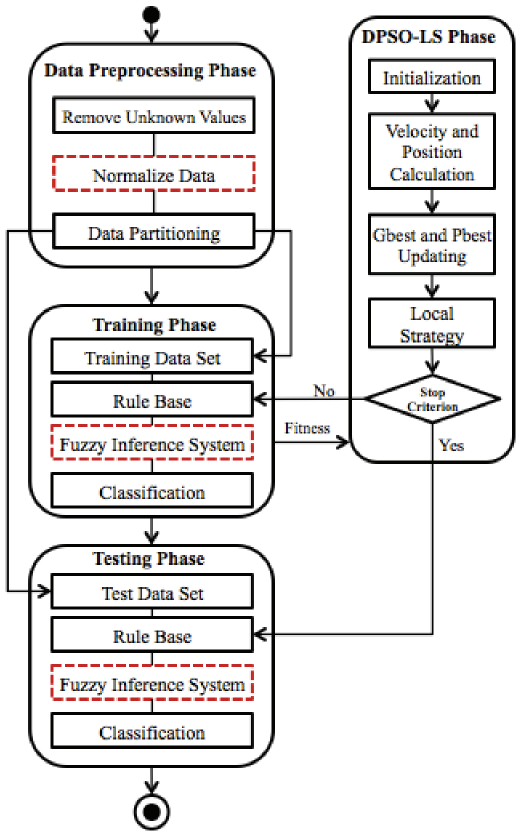

Figure 1. Processes of DPSO-LS based classifiers.

1) Data Preprocessing Phase: In this phase, firstly, we need to remove instances that have unknown values since the proposed system cannot handle these values. It is also known that the proposed system can only handle numerical data, if the class labels are non-numerical data, we convert it into numeric values. Then, the data set is randomly partitioned into 10 folds. 9 folds of the data is training data, which is used in the training phase, and 1 fold of the data is testing data, which is used in the testing phase.

2) DPSO-LS Phase: In this phase, the swarm is initialized. The velocity and position of each particle in the swarm are calculated. *GBest* and *Pbest* as described above are calculated, and their values are updated after the velocity and position have been updated accordingly. A local strategy is applied every 20 iterations. If the stopping criterion has not been met, *Gbest* and *Pbest* are forwarded to the training phase to calculate the overall fitness (see Equation 12), and individual fitness (see Equation 13). The DPSO-LS is stopped when the maximum number of iterations is met. The final $Gbest$ is forwarded to the testing phase.

3) Training Phase: A rule base which is forwarded by the DPSO-LS phase is used to classify the training data set. The overall fitness and individual fitness are calculated accordingly and are forwarded to the DPSO-LS.

4) Testing Phase: The final $Gbest$ forwarded by the DPSO-LS phase is used to classify the testing data set, and the experimental measures are calculated.

C. FDPSO-LS Classifier

A modified classifier, called fuzzy DPSO-LS classifier (FDPSO-LS), is implemented for both discrete and continuous data. A fuzzy partition with a simple fuzzy grid is adopted. Fuzzy set theory and the concept of linguistic variables, which were proposed by Zadeh [7], [8], have been widely used in pattern recognition and fuzzy reasoning. The use of the simple fuzzy partition method on classification rule discovery has been introduced in [21]. Applications on the fuzzy rule generation for control problems were proposed in [34]. Moreover, several fuzzy approaches for partitioning a pattern space were discussed in [35], [36]. Specially, for an example of using the simple fuzzy partition method in Fig. 2, each attribute can be partitioned into three linguistic terms (L = low, M = medium, H = high). Triangular membership functions are used for the linguistic terms. In the proposed method, each linguistic term is viewed as a candidate 1-dimension fuzzy grid. Considering a two-class classification problem as in Fig. 2, two antecedents with three membership functions can be partitioned into 9 grids on a 2-dimension plane. The closed circles and open circles denote the pattern in class 1 and class 2, respectively.

However, in the case of an n-dimensional classification problem, where each dimension has m linguistic terms, the possible number of rules is m^n. As the number of rules rises, an efficient algorithm that can automatically find the fuzzy rules is important and necessary.

Normally, several rules of the rule base are fired in the fuzzy rule classification system. The predicted class for a given instance is determined by the membership degree of the input variables. Specifically, for each class k,

$$\beta_{\text{Class k}} = \arg\max_{k} \sum_{1\leq i\leq n} \prod_{1\leq j\leq m} \mu_{ij} \tag{14}$$

where μ_{ij} is the input membership degree of the i^{th} rule of the j^{th} antecedent. The class that has the largest β value is selected as the predicted class. Moreover, unlike the DPSO-LS classifier, the rule that has the smallest β value is chosen as the worst rule.

As shown in Fig. 1, FDPSO-LS has similar processes as DPSO-LS, however, all the rectangles are used. The four main phases of data preprocessing, training, DPSO-LS and testing are similar to DPSO-LS. In the data preprocessing

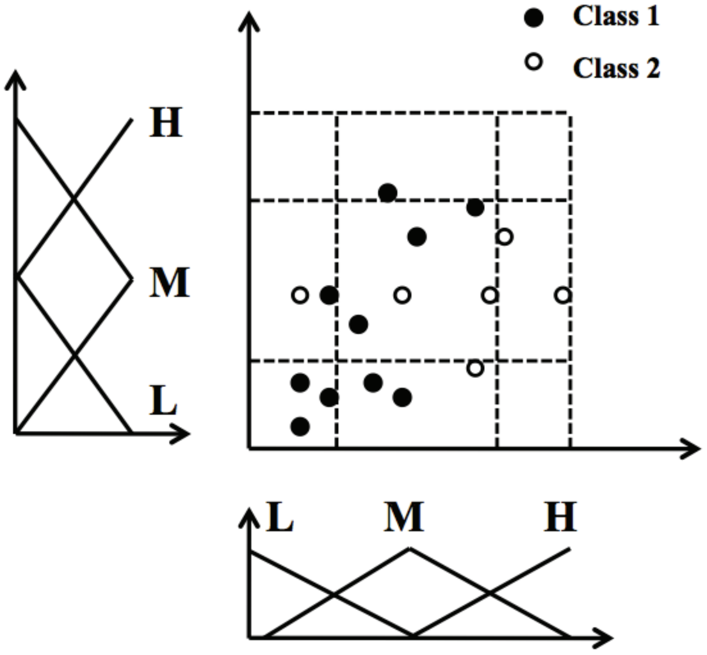

Figure 2. An example of fuzzy partition

phase, besides the removal of unknown values and data partitioning processes, a data normalization process is used to normalize continuous data. Each column of the data set is normalized between 0 and 1 using Equation 15:

$$X_i = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{15}$$

where X_i is the i^{th} value of the column. X_{min} is the minimum value of the column, and X_{max} is the maximum value of the column. The data set is partitioned into 10 folds. 9 folds of the data are used as the training data set, and the remainder is used as the test data set for the implementation.

The DPSO-LS phase is the same as for the DPSO-LS classifier. However, in the training and testing phases, a fuzzy inference process is added for the fuzzy reasoning process. The Fuzzy Inference System (FIS) is a popular computing system based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. It has been successfully applied to a wide variety of fields, such as automatic control, data classification, expert systems, decision analysis, etc. Due to its multidisciplinary nature, FIS is known by numerous other names. However, we only concentrate on the concept of the fuzzy IF-THEN rules.

The basic structure of a fuzzy inference process consists of three modules: fuzzification, fuzzy rule base and inference, and defuzzification. As shown in Fig. 3, a crisp input is taken, the fuzzification module coverts the crisp input into a fuzzy input using the fuzzy set theory. In the second module, fuzzy rules are contained in a rule base and a reasoning mechanism that performs the inference procedure is included. Finally, a method of defuzzification to extract a crisp output that represents a fuzzy set is needed by the third module. Due to the way outputs are determined, there are two types of inference systems: Mamdani and Sugeno. Mamdani's fuzzy inference system was among the first control systems built using fuzzy set theory, which was proposed in 1975 by Ebrahim Mamdani [37]. Sugeno, or Takagi-Sugeno-Kang, was introduced in 1985 [38]. It is similar to the Mamdani method in many respects, however, the main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant. In this approach, only the Mamdani style of defuzzification is considered.

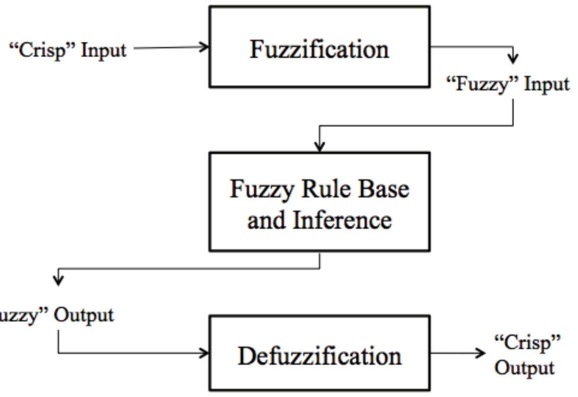

Figure 3. Fuzzy inference process.

IV. Experiments and Results

As mentioned above, the experiments are conducted for three approaches: DPSO (without local strategy), DPSO-LS and FDPSO-LS. The experimental setup for both approaches are described in the following subsection followed by the description of the experimental results of both approaches. The results of DPSO-LS and FDPSO-LS are listed, respectively, followed by a comparison.

A. *Experimental Setup*

The experiments of the two approaches are conducted on a number of datasets taken from the UCI repository [39]. The experiments of the two approaches are evaluated on an ASUS desktop (Intel(R) Dual Core I3 CPU @3.07 GHz, 3.07 GHz) Matlab Version 7.13. All measurements of the two approaches are tested 10 times using 10-fold cross validation [5]. Each data set is divided into 10 random partitions. Nine partitions of the data set are used as the training data, and one partition is selected as the test data.

B. *Results of the DPSO-LS Approach*

As far as the performance evaluation is concerned for the proposed DPSO-LS algorithm, a comparison with other rule classification algorithms JRip, PART and decision tree algorithm J48 is performed. These three algorithms have been implemented in WEKA (Waikato Environment for Knowledge Analysis) [5]. The algorithms are summarized as follows:

- **JRip** is a RIPPER rule learning algorithm [40]. JRip is based on association rules with reduced error pruning (REP), and integrates reduced error pruning with a separate-and-conquer strategy. It is a very common and effective technique found in decision tree algorithms.
- **PART** is created by Frank and Witten [41] for a partial decision tree. PART combines the separate-and-conquer strategy of RIPPER with the decision tree. It works by building a rule and removing its cover until all the instances are covered.
- **J48** is a decision tree implementation induced by the C4.5 algorithm, which is developed by Quinlan [4]. It learns decision trees for the given data by constructing them in a top-down way.

Table I
Parameters and their values of the DPSO and DPSO-LS algorithms.

Paramete	Values
Swarm Size	25
Maximum Iteration	100
(ω_1, ω_2, ω_3)	(0.2, 0.3, 0.5)
[V_{min}, V_{max}]	[-1, 1]

Table I shows the parameters and their values used for our DPSO, DPSO-LS and FDPSO-LS algorithms. Usually, a large swarm size requires less iterations to reach convergence in PSO. In our proposed algorithm, the swarm size is chosen as 25, and the maximum number of iterations for each run is set to 100. The description of the selected data sets used are summarized in terms of number of attributes, number of instances and number of classes as shown in Table II. The 6 data sets are listed alphabetically, where data set *Breast-L* and *Breast-W* are abbreviations for *Ljubljana Breast Cancer* and *Wisconsin Breast Cancer*, respectively. Measured are the rule size, the weighted average True Positive Rates (TPRs) and False Positive Rates (FPRs), as well as the precision.

Table II
Datasets used for the experiments.

Data Set	Attributes	Instances	Classes
Balance-scale	4	625	3
Breast-L	9	286	2
Breast-W	9	699	2
Car	6	1728	4
Lymphography	18	146	4
Tic-Tac-Toe	9	958	2

As we mentioned before, the DPSO can easily get stuck in local optima. In order to see the performance improvements of the local strategy, we compare DPSO (without local strategy) with DPSO-LS (with local strategy) by running them 10 times for 100 iterations each. The average coverage of the 10 runs is listed in TABLE III. A corresponding two-tailed Student's t-test with a significance level of 5% is applied. The results show that the proposed DPSO-LS can achieve better coverage in all cases. However, DPSO-LS only shows significant improvements in 3 of 6 cases.

Table III
AVERAGE COVERAGE OF DPSO AND DPSO-LS FOR 100 ITERATIONS.

Data Set	DPSO(%)	DPSO-LS(%)	Significance
Balance-scale	77.27 ± 3.72	**83.39** ± 3.20	**Yes**
Breast-L	82.57 ± 2.63	**86.71** ± 1.07	**Yes**
Breast-W	91.43 ± 4.25	**94.20** ± 4.30	No
Car	94.92 ± 5.06	**97.30** ± 4.40	No
Lymphography	76.23 ± 3.51	**80.10** ± 3.60	**Yes**
Tic-Tac-Toe	100.00 ± 0.00	100.00 ± 0.00	No

In Fig. 4, we see the coverage of DPSO-LS compared to DPSO, JRip, PART and J48. Error bars are shown on the histograms of the DPSO-LS and DPSO (for the other algorithms, no variants were reported since they are not captured by WEKA). In most cases, the DPSO-LS algorithm has a higher coverage. Although the Breast-W data set does not show better results, the values of the other four algorithms are very close.

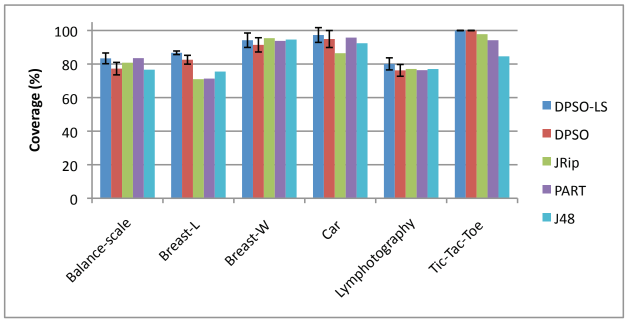

Figure 4. Coverage of all algorithms.

For all rule mining algorithms it is necessary to test the average rule set size to indicate the complexity of the rule set produced by each algorithm. Table IV lists the size of the rule set required for DPSO, DPSO-LS, JRip, PART, and J48. As shown in the table, the JRip algorithm always requires the least number of rules, while the PART algorithm requires the most number of rules. J48 uses by far the most number of rules with the exception of the Breast-L data set. The number of rules for the proposed DPSO-LS algorithm is less than the PART algorithm. Both DPSO-LS and DPSO show comparable results in terms of rule size.

Table V lists the average weighted TPR, which is also referred to as sensitivity. As shown in the table, the proposed

Table IV
AVERAGE RULE SIZE OF ALL ALGORITHMS.

Data Set	JRip	PART	J48	DPSO	DPSO-LS
Balance-scale	**12**	47	52	26.01±3.10	24.70±2.66
Breast-L	**3**	20	4	15.25±4.50	17.00±3.50
Breast-W	**6**	10	14	**6.05**±3.01	7.13±2.08
Car	49	68	131	**43.20**±5.20	44.18±4.17
Lymphography	**6**	13	21	11.15±2.50	9.40±3.06
Tic-Tac-Toe	**9**	49	95	35.3±3.76	38.80±1.70

algorithm, DPSO-LS, scores better than DPSO, JRip, PART and J48 in terms of sensitivity.

Table V
AVERAGE WEIGHTED TPRS (%) OF ALL ALGORITHMS.

Data Set	JRip	PART	J48	DPSO	DPSO-LS
Balance-scale	80.8	**87.5**	76.6	80.20±3.12	87.40±2.30
Breast-L	71.0	71.3	75.5	81.80±2.22	**89.50**±3.70
Breast-W	95.4	93.8	94.6	92.36±4.30	**97.27**±2.10
Car	86.5	95.8	92.4	93.50±3.10	**98.84**±1.33
Lymphography	77.7	76.4	77.0	73.30±3.26	**80.50**±4.40
Tic-Tac-Toe	97.8	94.3	84.6	**100.00**±0.00	**100.00**±0.00

The weighted average FRPs, which represent *1-Specificity*, are listed in Table VI. The FPRs of DPSO-LS are less than the other algorithms except for the Lymphography data set.

Table VI
AVERAGE WEIGHTED FPRS (%) OF ALL ALGORITHMS.

Data Set	JRip	PART	J48	DPSO	DPSO-LS
Balance-scale	16.4	9.7	17.3	15.01±3.25	**8.70**±2.20
Breast-L	48.9	54.2	52.4	25.80±4.30	**16.00**±7.20
Breast-W	4.4	8.0	6.4	1.20±0.18	**0.50**±0.01
Car	6.4	1.6	5.6	5.27±2.30	**1.04**±0.05
Lymphography	21.6	21	**18.7**	30.11±5.60	22.00±3.40
Tic-Tac-Toe	3.10	7.6	19.1	**0.00**±0.00	**0.00**±0.00

The weighted average precision values are compared in Table VII. The precision of the DPSO-LS is always better than DPSO, JRip, PART and J48, showing the largest improvement on the Breast-L data set.

Table VII
AVERAGE WEIGHTED PRECISION (%) OF ALL ALGORITHMS.

Data Set	JRip	PART	J48	DPSO	DPSO-LS
Balance-scale	74.5	83.3	73.2	79.95±3.80	**85.40**±3.20
Breast-L	68.8	68.2	75.2	83.16±3.30	**89.50**±3.60
Breast-W	95.5	93.8	94.6	92.35±4.10	**96.59**±2.15
Car	88.1	95.9	92.4	93.76±3.30	**99.10**±1.20
Lymphography	76.5	76.6	77.6	71.31±5.12	**78.57**±5.80
Tic-Tac-Toe	97.8	94.2	84.6	**100.00**±0.00	**100.00**±0.00

C. Comparison and Results of DPSO-LS and FDPSO-LS for Discrete Data Sets

In order to compare the performance of DPSP-LS and FDPSO-LS on discrete data sets, a corresponding two-tailed Student's t-test with a significance level of 5% is applied.

As shown in Table VIII, only 2 of the 5 data sets show significant improvements.

Table VIII
AVERAGE COVERAGE OF DPSO-LS AND FDPSO-LS IN 100 ITERATIONS.

Data Set	DPSO-LS(%)	FDPSO-LS(%)	Significance
Balance-scale	77.27 ± 3.72	77.13 ± 2.50	No
Breast-L	82.57 ± 2.63	86.71 ± 1.07	Yes
Breast-W	91.43 ± 4.25	93.20 ± 2.30	No
Car	94.92 ± 5.06	97.30 ± 4.40	No
Lymphography	76.23 ± 3.51	80.10 ± 3.60	Yes

In Fig. 5, we see the average coverage of the DPSO-LS compared to FDPSO-LS. Error bars are shown on the histograms of both the proposed algorithms. In most cases, the proposed FDPSO-LS algorithm has a higher coverage. Besides the Balance-scale data set, FDPSO-LS achieves better results for the other four data sets.

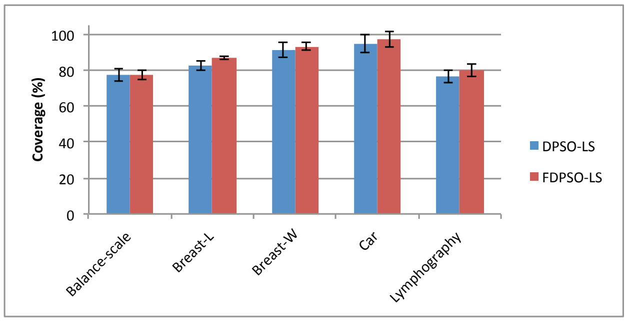

Figure 5. Coverage comparison of DPSO-LS and FDPSO-LS.

In Table IX, the average rule size of DPSO-LS and FDPSO-LS is compared. FDPSO-LS requires less number of rules than DPSO-LS due to the usage of the linguistic variables.

Table IX
AVERAGE RULE SIZE OF DPSO-LS AND FDPSO-LS IN 10 RUNS.

Data Set	DPSO-LS(%)	FDPSO-LS(%)
Balance-scale	24.70 ± 2.66	**7.12** ± 2.10
Breast-L	17.00 ± 3.50	**10.24** ± 3.07
Breast-W	7.13 ± 2.08	**7.08** ± 2.30
Car	44.18 ± 4.17	**14.12** ± 4.40
Lymphography	9.40 ± 3.06	**5.60** ± 3.60

Table X shows the average weighted TPR of DPSO and FDPSO-LS. FDPSO-LS does not show improvements compared to DPSO-LS for discrete data sets. FDPSO-LS scores slightly better on 2 out of 5 data sets.

As shown in Table XI, FDPSO-LS has a smaller FPRs in most cases except for the Car data set.

In terms of average weighted precision, FDPSO-LS does not show improvements compared to DPSO-LS on the discrete data sets except for Lymphography as shown in Table XII.

Table X
AVERAGE WEIGHTED TPRS OF DPSO-LS AND FDPSO-LS IN 10 RUNS.

Data Set	DPSO-LS(%)	FDPSO-LS(%)
Balance-scale	87.40 ± 2.30	**88.13** ± 3.12
Breast-L	**89.50** ± 3.70	86.71 ± 2.70
Breast-W	**97.27** ± 2.10	95.7 ± 2.30
Car	**98.84** ± 1.33	93.80 ± 3.45
Lymphography	80.50 ± 4.40	**83.80** ± 2.60

Table XI
AVERAGE WEIGHTED FPRS OF DPSO-LS AND FDPSO-LS IN 10 RUNS.

Data Set	DPSO-LS(%)	FDPSO-LS(%)
Balance-scale	8.70 ± 2.20	**1.65** ± 2.80
Breast-L	16.00 ± 7.20	**7.42** ± 1.07
Breast-W	5.00 ± 1.20	**4.10** ± 2.23
Car	**1.04** ± 0.05	5.80 ± 2.18
Lymphography	22.00 ± 3.4	**16.50** ± 2.34

Overall, with respect to discrete data sets, FDPSO-LS does not significant good improvements in most cases. One reason is that it does not efficiently normalize discrete data sets using liguistic terms. Usually, it causes overfitting and decreases the accuracy. For example, for the Balance-scale data set each attribute has either 3 or 4 discrete values, and FDPSO-LS uses 3 membership functions. When we normalize the attribute values into 3 membership function, the data does not partition well for the attributes having small discrete values.

D. Results of FDPSO-LS Approach for Continuous Data Set

As far as the performance evaluation for the proposed FDPSO-LS is concerned, a comparison with other rule classification algorithm FURIA is performed. FURIA is short for Fuzzy Unordered Rule Induction Algorithm which extends the well-known RIPPER algorithm [42]. FURIA learns unordered fuzzy rule sets instead of rule lists. It includes a number of modifications and extensions to deal with uncovered examples.

The description of the selected data sets used are summarized in terms of number of attributes, number of instances, and number of classes as shown in Table XIII. The 5 data sets are listed alphabetically.

Measured are also the rule size evolved, the weighted average TPRs and FPRs, as well as the precision.

In order to observe the performance, we compared FURIA with FDPSO-LS by running both algorithms 10 times for 100 iterations each. The average coverage of the 10 runs is listed in TABLE XIV. The corresponding two-tailed Student's t-test with a significance level of 5% was applied. The results show that the proposed FDPSO-LS can achieve better coverage in most cases except for the glass data set. However, FDPSO-LS only shows significant improvements for 2 of the 5 data sets.

In Fig. 6, we see the average coverage of the proposed FDPSO-LS compared to FURIA. Error bars are shown on the histograms of the proposed FDPSO-LS. For most

Table XII
AVERAGE WEIGHTED PRECISION OF DPSO-LS AND FDPSO-LS IN 10 RUNS.

Data Set	DPSO-LS(%)	FDPSO-LS(%)
Balance-scale	**85.40** ± 3.20	82.10 ± 2.50
Breast-L	**89.50** ± 3.60	85.71 ± 3.32
Breast-W	96.59 ± 2.15	**96.70** ± 4.30
Car	**99.10** ± 1.20	97.30 ± 2.40
Lymphography	78.57 ± 5.80	**82.80** ± 3.41

Table XIII
DATASETS USED FOR THE PROPOSED FUZZY RULE-BASED SYSTEM USING DPSO-LS.

Data Set	**Attributes**	**Instances**	**Classes**
Breast-W	9	699	2
Glass	10	214	7
Haberman's Survival	3	306	2
Iris	4	150	3
Pima Indians Diabetes	8	768	2

data sets, the proposed FDPSO-LS algorithm has a higher coverage. Besides, for the glass data set FDPSO-LS obtains higher coverage for the other data sets.

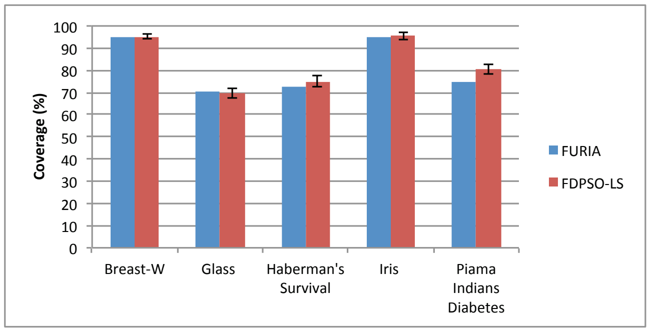

Figure 6. Coverage comparison of the proposed FDPSO-LS and FURIA.

Table XV lists the size of the rule set required for FDPSO-LS and FURIA. As shown in the table, the number of rules for the proposed FDPSO-LS is less than for the FURIA algorithm for most data sets. The reason is that the proposed FDPSO-LS reduces the rule size since it uses the local strategy. The values after ± are standard deviations of the corresponding results.

Table XVI lists the average weighted True Positive Rates (TPRs), which are also referred to as sensitivity. As shown in the table, the proposed algorithm, FDPSO-LS, scores better than FURIA for most data sets in terms of sensitivity except for the Glass data set.

The weighted average FPRs, which represent *1-Specificity*, are listed in Table XVII. The FPRs of the proposed FDPSO-LS are less than FURIA, which indicates that FURIA has a higher false positive rate.

The weighted average precision values are compared in Table XVIII. The precision of FDPSO-LS is always better than FURIA, showing the largest improvement on the Haberman's Survival and Pima Indians Diabetes data sets.

Table XIV
AVERAGE COVERAGE OF FURIA AND FDPSO-LS FOR 100 ITERATIONS.

Data Set	FURIA(%)	FDPSO-LS(%)	Significance
Breast-W	94.71	**95.20**±1.30	No
Glass	70.56	69.70±2.20	NO
Haberman's Survival	72.55	**75.02**±2.40	**Yes**
Iris	94.67	**95.56**±1.70	No
Pima Indians Diabetes	74.48	**80.60**±2.30	**Yes**

Table XV
AVERAGE RULE SIZE OF FDPSO-LS AND FURIA.

Data Set	FURIA(%)	FDPSO-LS(%)
Breast-W	15	**7.12**±2.10
Glass	16	**9.40**±3.20
Haberman's Survival	**4**	7.20±2.40
Iris	5	**4.00**±1.70
Pima Indians Diabetes	**5**	7.70±2.30

V. CONCLUSION

In this study, we have proposed two classifiers: DPSO-LS and FDPSO-LS. Both classifiers are based on the proposed DPSO-LS algorithm, which uses a rule base to represent a 'particle' that evolves the rule base over time. DPSO-LS is implemented as a matrix of rules, representing IF-THEN classification rules, that have conjunctive antecedents and one consequent. In addition, a local mutation search strategy was incorporated in order to take care of the premature convergence of PSO. The DPSO-LS classifier was applied on discrete data sets based on the IF-THEN classification rules, while the FDPSO-LS is based on the concept of fuzzy IF-THEN rules and is applied to both discrete and continuous data sets.

Experiments were conducted using 6 discrete data sets and 5 continuous data sets that are taken from the UCI repository. Our DPSO-LS algorithm was compared against DPSO, JRip, PART and J48. In addition, FDPSO-LS was compared against FURIA. Measures used were rule size, TPRs, FPRs, and precision. The experimental results revealed that DPSO-LS achieves better performance for most data sets than FPSO-LS applied to discrete data sets. On the other hand, FDPSO-LS obtains better performance when applied to continuous data sets compared to FURIA.

As for future work, it would be interesting to compare the Pittsburgh approach with the Michigan approach. Moreover, FDPSO-LS has shown improved results compared to FURIA, which could still be improved by minimizing the number of rules and deleting replicated rules. Furthermore, the proposed FDPSO-LS can be further improved by applying discrete data sets with larger ranges of attribute values.

REFERENCES

[1] J. Jackson, Data Mining; A Conceptual Overview, Communications of the Association for Information Systems: Vol. 8, Article 19, 2002.

Table XVI
AVERAGE WEIGHTED TPRs OF FDPSO-LS AND FURIA.

Data Set	FURIA(%)	FDPSO-LS(%)
Breast-W	94.7	**95.20**±2.13
Glass	**70.6**	69.50±3.21
Haberman's Survival	72.5	**78.10**±2.72
Iris	94.7	**94.74**±2.33
Pima Indians Diabetes	74.5	**81.20**±3.11

Table XVII
AVERAGE WEIGHTED FPRs OF FDPSO-LS AND FURIA.

Data Set	FURIA(%)	FDPSO-LS(%)
Breast-W	6.7	**4.70**±3.23
Glass	13.1	**7.80**±3.81
Haberman's Survival	57.3	**33.80**±5.20
Iris	2.7	**2.58**±1.20
Pima Indians Diabetes	36.7	**23.30**±3.70

Table XVIII
AVERAGE WEIGHTED PRECISION OF FDPSO-LS AND FURIA.

Data Set	FURIA(%)	FDPSO-LS(%)
Breast-W	94.7	**96.70**±2.70
Glass	**70.5**	70.10±3.20
Haberman's Survival	69 .0	**77.80**±3.13
Iris	94.7	**95.10**±2.40
Pima Indians Diabetes	73.7	**80.23**±3.61

[2] M. Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms, IEEE Press & John Wiley, November 2002.

[3] J.R. Quinlan, Induction of Decision Trees, Mach. Learn. 1, 1 (Mar. 1986), 81-106, 1986.

[4] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

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

[6] P. Domingos and M.J. Pazzan, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning 29(2-3): 103-130 (1997).

[7] L.A. Zadeh, Fuzzy sets, Information and Control, Vol. 8, pp. 338-353, 1965.

[8] L. A. Zadeh, The concept of linguistic variable and its application to approximate reasoning, Inf Sci, 8 (1975).

[9] M. Chen and S.A. Ludwig, Discrete Particle Swarm Optimization With Local Search Strategy for Rule Classification, Proceeding of Fourth World Congress on Nature and Biologically Inspired Computing (IEEE NaBIC), Mexico City, Mexico, November 2012.

[10] M.V. Fidelis, H.S. Lopes, A.A. Freitas and P. Grossa, Discovering comprehensible classification rules with a genetic algorithm, In Proceedings of the 2000 Congress on Evolutionary Computation, La Jolla, CA, USA, IEEE, vol. 1, pp. 805-810.

[11] D.R. Carvalho and A.A. Freitas, A genetic-algorithm for discovering small-disjunct rules in data mining, Applied Soft Computing, vol. 2, pp. 75-88.

[12] K.K. Bharadwaj and B.M. Al-Maqaleh, Evolutionary approach for automated discovery of censored production rules, In: Proceedings of the 8th International Conference on Cybernetics, Informatics and Systemics (CIS-2005). vol. 10, Krakow, Poland, pp. 147-152, 2005.

[13] K.K. Bharadwaj and B.M. Al-Maqaleh, Evolutionary approach for automated discovery of augmented production rules, International Journal of Computational Intelligence. vol. 3, Issue 4, pp. 267-275, 2006.

[14] S. Yogita and D. Kumar, Rule Exceptions: Automated discovery of comprehensible decision Rules, IEEE International Advance Computing Conference (IACC2009), Patiala, India, pp. 1479-1483,2009.

[15] H. Ishibuchi, and Y. Nojima. Evolutionary multiobjective optimization for the design of fuzzy rule-based ensemble classifiers. International Journal of Hybrid Intelligent Systems 3.3 (2006): 129-145.

[16] R.S. Parepineslli, H.S. Lpes and A. Freitas. An Ant Colony Algorithm for classification Rule Discovery. Data Mining: a Heuristic Approach,Idea Group Publishing, 2002.

[17] N. Holden and A.A. Freitas, A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining, Journal of Artificial Evolution and Applications, vol. 2008, Article ID 316145, 11 pages, 2008.

[18] Z. Wang, X. Sun and D. Zhang, A PSO-Based Classification Rule Mining Algorithm, In proceedings of the 3rd International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications. 2007.

[19] H. Ishibuchi, T. Murata, and I.B. Turksen, Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems, Fuzzy Sets and Systems, vol. 89, pp. 135150, 1997.

[20] C.H. Wang, T.P. Hong, and S.S. Tseng, Integrating fuzzy knowledge by genetic algorithms, Fuzzy Sets and Systems, vol. 2, no. 4, pp. 138149, 1998.

[21] H. Ishibuchi, T. Nakashima, and T. Murata, Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, vol. 29, no. 5, pp. 601618, 1999.

[22] M. Russo, FuGeNeSys- a fuzzy genetic neural system for fuzzy modeling, IEEE Transactions on Fuzzy Systems, vol. 6, no. 3, pp. 373C388, 1998.

[23] L. Wang and J. Yen, Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and Kalman filter, Fuzzy Sets and Systems, vol. 101, pp. 353C362, 1999.

[24] Y. Shi, R. Eberhart, and Y. Chen, Implementation of evolutionary fuzzy systems, IEEE Transactions on Fuzzy Sytems, vol. 7, no. 2, pp. 109C119, 1999.

[25] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy system: Evolutionary tuning and learning of fuzzy knowledge bases, In: Advances in fuzzy systemsapplications and theory, vol 19. World Scientific, Singapore, 2001.

[26] M.F. Ganji, M.S. Abadeh, Using fuzzy ant colony optimization for diagnosis of diabetes disease, Electrical Engineering (ICEE), 2010 18th Iranian Conference on, vol., no., pp.501-505, 11-13 May 2010 doi: 10.1109/IRANIANCEE.2010.

[27] M.F. Ganji, M.S. Abadeh, Parallel fuzzy rule learning using an ACO-based algorithm for medical data mining, In Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on (pp. 573-581). IEEE.

[28] C.N. Raju, O.R. Devi, S. Mani and S. Nagendram, An Improved Ant Colony Optimization Technique by using Fuzzy Inference Rules for Image Classification and Analysis, In Interna-tional Journal of Advanced Engineering & Application, pp. 230-234, 2010.

[29] R.P. Prado, S.G. Galan, J. E. Exposito and A. J. Yuste, Knowledge Acquisition in Fuzzy-Rule-Based Systems With Particle-Swarm Optimization, IEEE T. Fuzzy Systems 18(6): 1083-1097 (2010).

[30] F. Beloufa and M. A. Chikh, Automatic Fuzzy rules-base generation using Modified Particle Swarm Optimization, 2012.

[31] M.S. Abadeh, J. Habibi, S. Aliari, Using a particle swarm optimization approach for evolutionary fuzzy rule learning: a case study of intrusion detection, Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU06), Paris, France, 27 July 2006 (2006)

[32] A. Cervantes, I.M. Galvan, P. Isasi, A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm, Congress on Evolutionary Computation 2005: 290-297.

[33] R.C. Eberhart and J. Kennedy, A New Optimizer using Particle Swarm Theory, In Proc. 6th Symp. Micro Machine and Human Science, Nagoya, Japan 1995, 29-43.

[34] L.X. Wang, J. M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on Systems, Man, and Cybernetics 22(6) (1992) 14141427.

[35] C.T. Sun, Rule-base structure identification in an adaptive-networkbased fuzzy inference system, IEEE Transactions on Fuzzy Systems 2(1) (1994) 6473.

[36] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[37] E.H. Mamdani, and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13, 1975.

[38] M. Sugeno, Industrial applications of fuzzy control, Elsevier Science Pub. Co., 1985.

[39] A. Frank & A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[40] W. Cohen, Fast effective rule induction, In Proceedings of Twelfth International Conference on Machine Learning, pages 115123, Tahoe City, CA, July 1995.

[41] E. Frank and I. H. Witten, Generating accurate rule sets without global optimization, In Proc. of the Intl Conf. on Machine Learning, pages 144151. Morgan Kaufmann Publishers Inc., 1998.

[42] J.C. Huhn, E. Hullermeier, FURIA: an algorithm for unordered fuzzy rule induction, Data Min. Knowl. Discov. 19(3): 293-319 (2009)