Discrete Particle Swarm Optimization With Local Search Strategy 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*—Rule discovery is an important classification method that has been attracting a significant amount of researchers in recent years. Rule discovery or rule mining uses a set of IF-THEN rules to classify a class or category. 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 implementation with a local search strategy (DPSO-LS) was devised. The local search strategy helps to overcome local optima in order to improve the solution quality. Our DPSO-LS uses the Pittsburgh approach whereby a rule base is used to represent a 'particle'. This rule base is evolved over time as to find the best possible classification model. Experimental results reveal that DPSO-LS outperforms other classification methods in most cases based on rule size, TP rates, FP rates, and precision.**

***Keywords*-Rule classification, 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. Given this tremendous amount of data, efficient and effective tools need to be present to analyze and reveal valuable knowledge that is hidden. The objective of the field of knowledge discovery and data mining are the discovery of knowledge that is not only correct, but also comprehensible.

Several data mining tasks have emerged that include classification, clustering, regression, dependence modeling, etc. The classification task is characterized by the organization of data into given 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 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. [1]. 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 [2]. However, since then, ID3 has been further improved and is now known as C4.5 [3]. 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 the 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 for the underlying model [4]. 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 assumptions [5].

This paper introduces a rule-mining algorithm that is based on a biologically-inspired algorithm. In particular, a particle swarm optimization approach is used to extract rules as a model for the prediction of classes.

The remainder of the paper is organized as follows. In Section II, related work is discussed. Our proposed algorithm is introduced in detail in Section III. The experimental setup and results are described in Section IV. The conclusion and future work is discussed in Section V.

II. Related Work

Related work in classification rule mining using biology-inspired algorithms mainly include evolutionary algorithms and swarm intelligent approaches. Genetic algorithm (GA) based concept learning uses either the Pittsburgh approach or the Michigan approach [6]. 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 on the other hand, 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 [6], 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 [7]. 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 [8], [9], [10], 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.

With regards to swarm intelligence approaches, a classification algorithm called Ant-Miner, first introduced in [11], has been applied successfully on rule classification problems. Particle Swarm Optimization (PSO) is another approach inspired by nature. However, most of the swarm intelligence algorithms for rule classification are based on the Michigan approach ([13], [15]).

PSO has been proven to be able to achieve a faster convergence than the GA algorithm [16]. In addition, it has been experimentally shown that the PSO algorithm scales well and is not highly sensitive to the population size [16]. As far as the authors' knowledge is concerned, due to the lack of flexibility of the Pittsburgh approach [14], the Pittsburgh-based PSO algorithm on rule classification is rarely used in literature. On the other hand, the Michigan approach usually requires some changes in the definition of the PSO algorithm to repel a particle from its neighbor [14] in order to avoid convergences of particles. In addition, the Michigan approach aims to optimize each rule's quality individually, and does not take the interaction between other rules [15] into the account. In [16], 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. The results of the Pittsburgh approach outperformed other strategies such as GA. In the light of its simplicity and capability to control convergence, the Pittsburgh-based PSO is a promising strategy that can outperform most of the classical classification algorithms. For this purpose, a discrete particle swarm optimization (DPSO-LS) implementation is investigated with a local strategy for solving the classification problem. In addition, since PSO is known to get easily stuck in local optima, an improvement with a local strategy is introduced.

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

PSO was introduced by Eberhart and Kennedy [12] 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 in the 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 the historical information to steer towards local or global optima.

Specifically, a Discrete Particle Swarm Optimization (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 that have conjunctive antecedents and one consequent. Hence, the i^{th} particle is presented as follows:

$$P_i = \left| \begin{array}{ccccc} a^i_{1,1} & a^i_{1,2} & \dots & a^i_{1,n} & c^i_1 \\ a^i_{2,1} & a^i_{2,2} & \dots & a^i_{2,n} & c^i_2 \\ \dots & \dots & \dots & \dots & \dots \\ a^i_{m,1} & a^i_{m,2} & \dots & a^i_{m,n} & c^i_m \end{array} \right| \quad (1)$$

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. The antecedents are denoted as feature variables while the consequent is the class labels. Thus, a particle consists of m rules, where each rule has n antecedents and 1 consequent.

The values of every antecedent, which are feature values of the variable, is enumerated consecutively starting from 1. For instance, if an antecedent has 5 discrete feature values, it will be enumerated as $\{1, 2, 3, 4, 5\}$. In this way, 0 means the antecedent can be any value in the rule, or in other words, 0 indicates the variable absence in the rule. 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:

$$a^i_{j,k} \in [0, NF_{in}], j \in \{1, 2, ..., m\} \quad (2)$$

$$k \in \{1, 2, ..., n\} \quad (3)$$

$$c^i_j \in [1, NF_{out}] \quad (4)$$

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:

$$V_i = \begin{vmatrix} v^i_{1,1} & v^i_{1,2} & \dots & v^i_{1,n} & v^i_{1,n+1} \\ v^i_{2,1} & v^i_{2,2} & \dots & v^i_{2,n} & v^i_{2,n+1} \\ \dots & \dots & \dots & \dots & \dots \\ v^i_{m,1} & v^i_{m,2} & \dots & v^i_{m,n} & v^i_{m,n+1} \end{vmatrix} \tag{5}$$

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 of the velocity matrix.

$$\vec{V_i} = \begin{vmatrix} \hat{v}^i_{1,1} & \hat{v}^i_{1,2} & \dots & \hat{v}^i_{1,n} & \hat{v}^i_{1,n+1} \\ \hat{v}^i_{2,1} & v^i_{2,2} & \dots & v^i_{2,n} & v^i_{2,n+1} \\ \dots & \dots & \dots & \dots & \dots \\ \hat{v}^i_{m,1} & \hat{v}^i_{m,2} & \dots & \hat{v}^i_{m,n} & \hat{v}^i_{m,n+1} \end{vmatrix} \tag{6}$$

The values of $\vec{V_i}$ are randomly assigned by 1, 2 and 3, where 1, 2 and 3 are denoted as three directions. 1 is denoted as the direction of the particle's move from the current position to the local best position. 2 is denoted as the direction of the particle's move toward from current position to the global best position. 3 is denoted as the direction of the particle's move from the current position to other position randomly within a 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 7, 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. In our specific case, the local best position is the local best rule base that has been found by the particle so far. The global best position is the best rule base that has been found through iterations. Precisely, as shown in Equation 8, for the i^{th} particle, $V_1(t)$ is the difference between the local best position and the current position, while $\vec{V_i}$ are assigned to 1, 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 $\vec{V_i}$ are assigned to 2. Values of $V_3(t)$ are randomly assigned within the range (see Equation 9), while values of $\vec{V_i}$ are assigned to 3.

$$\omega_1 + \omega_2 + \omega_3 = 1 \tag{7}$$

$$V(t+1) = V_1(t) \oplus V_2(t) \oplus V_3(t) \tag{8}$$

$$V_3(t) \in [V_{min}, V_{max}] \tag{9}$$

where $\oplus$ denotes an addition between matrices. After the velocity matrix has been calculated, the particle position can be computed as follows:

$$P(t+1) = P(t) \oplus V(t+1) \tag{10}$$

Thus, the process is repeated until it reaches the desired number of iterations. The historical information, such as global best position and local best position, is stored. It has to be mentioned that the particle size remains unchanged through the whole optimization process. The size of a rule base is known apriori from other existing algorithms [16] and set accordingly. This is also the case for the Michigan approach where the size of a rule set need to be predefined. In addition, it has to be highlighted that the proposed algorithm does not face weight modification through the optimization of the rule base. Finally, each particle is updated using Equation 10. Hence, the range of constraints, as given in Equations 2-4, are considered within the update process.

A. Definition of Overall Fitness

We propose a rule selection method where the number of classification rules included in each rule set is fixed as 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:

$$F_{overall}(S) = Coverage = \frac{NCP(S)}{|S|} \tag{11}$$

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 training data set S. $F_{overall}$ is defined to measure the overall performance of a rule base as a solution that can correctly classify training patterns. The higher the value of the coverage, the better the solution.

B. Local Mutation Strategy

Since PSO in general can get easily stuck in a local optima, a local strategy need to be devised that is run after certain number of iterations have passed. In particular, the local strategy that was devised for DPSO-LS makes use of mutation. Since a 'particle' is a rule base that has a good performance there still might exist some bad rules. In the light of overcoming the shortage of the Pittsburgh approach with poor quality individuals and to avoid getting stuck in local optima, the proposed local strategy modifies the worst rule of the best rule base, that is 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 has improved 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 [15] equation, which is given below:

$$f_{local} = \frac{1 + TP}{1 + TP + FP} \tag{12}$$

where TP is the number of true positives, and FP is the number of false positives. Specifically, TP is the number of instances that match both the rule antecedent and the rule consequent. FP is the number of instances that match the rule antecedent, but do not match the consequent. After we have obtain the best rule base (the global best position), a pruning process is applied to prune the bad rules from the rule base. We use Equation 12 to prune the bad rules when the f_{local} values are less than 0.1 in every run.

C. Algorithm Steps

The steps of the proposed DPSO-LS algorithm can be described as follows:

- ***Step 1:*** Initialization phase - The input data set needs to be split into training set and test set. The parameters of the DPSO-LS algorithm need to be initialized. A set of particles are initialized randomly. In the phase of initialization, each rule base is initialized by assigning values to half of the variables. That is, only half of the antecedents are randomly assigned features values, the rest of the antecedents are assigned value 0.
- ***Step 2:*** Fitness evaluation - We use Equation 11 to measure the goodness of a particle in the swarm.
- ***Step 3:*** Local and global best position computation - A local best position of every particle is selected with the best fitness values that have been recorded so far. The best position of the swarm is compared with the current global best position, if it is greater than the global best values, the global position is updated with the best position found so far.
- ***Step 4:*** Velocity update - The velocity of each particle is updated by Equation 8. As mentioned before, the velocity is updated by considering the personal best position, global best position and random changes.
- ***Step 5:*** Position update - The position of each particle is updated simply according to Equation 10.
- ***Step 6:*** Mutation - For every 20 iterations, the worst rule of the global best position is selected and the antecedents of the selected rule are mutated 10 times at random.
- ***Step 7:*** Iteration - If the stopping criterion has not been met, go back to Step 2. Otherwise, go to Step 8.
- ***Step 8:*** Output and pruning - The best global position is generated as an output. The poor quality rules of the output are pruned if the fitness value of that rule is less than 0.1.
- ***Step 9:*** Testing - The pruned output from Step 8 is used to classify the test set in order to measure the performance.

IV. Experiments and Results

A. Experimental Setup

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. The algorithms are summarized as follows:

- **JRip** is a RIPPER rule learning algorithm [18]. 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 [17] 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 [3]. It learns decision trees for the given data by constructing them in a top-down way.

The experiments are conducted on a number of datasets taken from the UCI repository. The experiments are implemented on an ASUS desktop (Intel(R) Dual Core I3 CPU @3.07 GHz, 3.07 GHz) of Matlab Version 7.13. All measurements are tested 10 times using 10-fold cross validation [4]. Each data set is divided into 10 partitions. Nine partitions of the data set are used as the training data and one partition is selected as the test data.

Parameter	Values
Swarm Size	25
Maximum Iteration	100
$(\omega_1, \omega_2, \omega_3)$	(0.2, 0.3, 0.5)
$[V_{min}, V_{max}]$	[-1, 1]

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

Table I shows the parameters and their values used for our DPSO-LS algorithm. Normally, a large swarm size requires less iterations to reach a 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

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

Table II
Datasets used for the experiments.

rule size, the weighted average TP rates and FP rates, as well as the precision.

B. Comparison and Results of the Different Algorithms

As we have 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) and 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. The 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 the cases. However, DPSO-LS only shows significant improvements in 3 of 6 cases.

Data Set	DPSO(%)	DPSO-LS(%)	Significant
Balance-scale	77.27 ± 3.72	83.39 ± 3.2	Yes
Breast-L	82.57 ± 2.63	86.71 ± 1.07	Yes
Breast-W	91.43 ± 4.25	94.2 ± 4.3	No
Car	94.92 ± 5.06	97.3 ± 4.4	No
Lymphography	76.23 ± 3.51	80.1 ± 3.6	Yes
Tic-Tac-Toe	100 ± 0.0	100 ± 0.0	No

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

In Figure 1, 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). In most of the cases, the DPSO-LS algorithm has a higher coverage. Although the Breast-W data set does not show a better result, the values of the other four algorithms are very close.

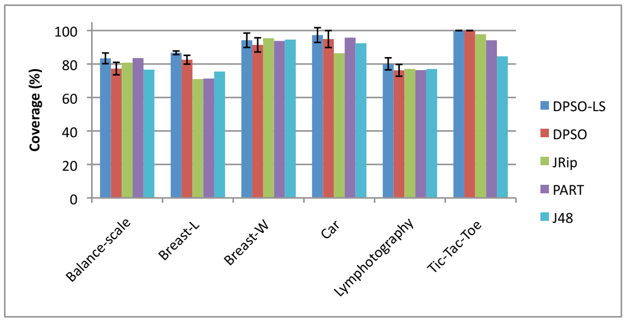

Figure 1. Coverage of all algorithms.

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-LS, DPSO, 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.

Data Set	DPSO-LS	DPSO	JRip	PART	J48
Balance-scale	24.7 ± 2.66	26.01 ± 3.1	12	47	52
Breast-L	17 ± 3.5	15.25 ± 4.5	3	20	4
Breast-W	7.13 ± 2.08	6.05 ± 3.01	6	10	14
Car	44.18 ± 4.17	43.2 ± 5.2	49	68	131
Lymphography	9.4 ± 3.06	11.15 ± 2.5	6	13	21
Tic-Tac-Toe	38.8 ± 1.7	35.3 ± 3.76	9	49	95

Table IV
AVERAGE RULE SIZE OF ALL ALGORITHMS.

Table V lists the average weighted true positive rates (TP rates), which is also referred to as sensitivity. As shown in the table, the proposed algorithm, DPSO-LS, scores better than DPSO, JRip, PART and J48 in terms of sensitivity.

Data Set	DPSO-LS	DPSO	JRip	PART	J48
Balance-scale	87.4 ± 2.3	80.2 ± 3.12	80.8	87.5	76.6
Breast-L	89.5 ± 3.7	81.8 ± 2.22	71	71.3	75.5
Breast-W	97.27 ± 2.1	92.36 ± 4.3	95.4	93.8	94.6
Car	98.84 ± 1.33	93.5 ± 3.1	86.5	95.8	92.4
Lymphography	80.5 ± 4.4	73.3 ± 3.26	77.7	76.4	77
Tic-Tac-Toe	100 ± 0.0	100 ± 0.0	97.8	94.3	84.6

Table V
AVERAGE WEIGHTED TP RATES (%) OF ALL ALGORITHMS.

The weighted average false positive rates (FP rate), which represent *1-Specificity*, are listed in Table VI. The FP rates of DPSO-LS are less than the other algorithms except for the Lymphography data set.

Data Set	DPSO-LS	DPSO	JRip	PART	J48
Balance-scale	8.7 ± 2.2	15.01 ± 3.25	16.4	9.7	17.3
Breast-L	16.0 ± 7.2	25.8 ± 4.3	48.9	54.2	52.4
Breast-W	0.5 ± 0.01	1.2 ± 0.18	4.4	8.0	6.4
Car	1.04 ± 0.05	5.27 ± 2.3	6.4	1.6	5.6
Lymphography	22 ± 3.4	30.11 ± 5.6	21.6	21	18.7
Tic-Tac-Toe	0	0	3.1	7.6	19.1

Table VI
AVERAGE WEIGHTED FP RATES (%) OF ALL ALGORITHMS.

The weighted average precisions 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.

V. CONCLUSION

In this study, we have introduced a Pittsburgh-based discrete particle swarm optimization algorithm for rule mining, abbreviated as DPSO-LS. DPSO-LS uses a rule base to represent a 'particle' and 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

Data Set	DPSO-LS	DPSO	JRip	PART	J48
Balance-scale	85.4±3.2	79.95±3.8	74.5	83.3	73.2
Breast-L	89.5±3.6	83.16±3.3	68.8	68.2	75.2
Breast-W	96.59±2.15	92.35±4.1	95.5	93.8	94.6
Car	99.1±1.2	93.76±3.3	88.1	95.9	92.4
Lymphography	78.57±5.8	71.31±5.12	76.5	76.6	77.6
Tic-Tac-Toe	100±0.0	100±0.0	97.8	94.2	84.6

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

search strategy was incorporate in order to take care of the premature convergence of PSO.

The experiments were conducted using 6 data sets that are taken from the UCI repository. Our DPSO-LS algorithm was compared against DPSO, JRip, PART and J48. Measures used were rule size, TP rates, FP rates, and precision. The experimental results revealed that DPSO-LS achieves a better performance in most cases than the other approaches based on the aforementioned measures.

As for future work, since the Pittsburgh approach suffers from bad quality rules within the rule set when only the overall performance is considered, the quality of each rule is not taken into account. Thus, it would be interesting to compare the Pittsburgh approach with the Michigan approach. Moreover, fuzzy rule learning is another approach that would be an interesting direction to explore.

REFERENCES

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

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

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

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

[5] 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).

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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).

[17] 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.

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