I J C T A, 9(3), 2016, pp. 1495-1506

GSOMINER: A Supervised Classification Rule Miner based on Group Search Optimization

K. Joshil Raj* and S. Siva Sathya**

Abstract: Group Search Optimization (GSO) is a novel bio inspired technique derived from the interactions of individuals of a population while searching for food. It is inspired by the searching behavior of animals and this strategy has been used to design optimum strategies for search of food (continuous optimization problems). This paper proposes the generation of classification rules using GSO for supervised classification. The methodology proposed in the paper is to extract classification rules using the searching strategies of GSO and apply the same on a given dataset so as to achieve accurate classification of the data in to requisite classes as part of supervised classification. In doing so, the aim was to generate a set of classification rules that give higher classification accuracy. The quality and effectiveness of the proposed methodology has been evaluated on standard machine learning datasets to obtain classification rules**.**

Keywords: Evolutionary algorithm; Group Search Optimizer; Machine learning; Rule Mining; Swarm Intelligence; Particle Swarm Optimisation; Ant Colony Optimisation

1. INTRODUCTION

Classification rule generation is an emerging field of data classification with algorithms generating effective set of rules from the dataset to efficiently classify the data into predetermined classes. In classification rule generation, a classification algorithm is applied on the training dataset having class information for each record and rules are generated using the correlation between the values of the attributes and the class information. These generated rules are then further used to predict the class information for the records of the test dataset. The classification rules generated from the training dataset are depicted in the form of IF-THEN rules that represent knowledge in a simple and comprehensible form [1]. The generated rules can be further evaluated or compared based on several factors like number of rules generated, number of members in each rule, classification accuracy derived by using the rules, simplicity and comprehensibility of the rules etc. Classification rule generation is an important and emerging field of data mining that functions by generating simple set of symbolic rules to describe each class of the dataset in terms of values of attributes in a natural way [2].

The generated rules are depicted in the form of *IF <attribute_values> THEN <class_information>* where in *<attribute_values>* represents the condition part of the rule that consists of logical combination of different values of the attributes of the dataset (for instance: *attribute_values 1 AND attribute_values 2 AND...*) and *THEN <class_information>* is the outcome of the rule giving out the predicted class information by that specific rule for all those records, instances or elements of the dataset which satisfy the conditions of the rule. Each term in the generated rule is a triple i.e. it consists of the logical combination of *<attribute_value, operator, class_information>*, where the *operator* is a relational operator that combines multiple attribute values together to generate the outcome in terms of the class information.

This paper describes an attempt to use the population searching strategy technique based optimization algorithm, Group Search Optimizer (GSO) [3] for generation of rules for classification. Each GSO member

* Department of Computer Science, Pondicherry University, Pondicherry, India, *Email: joshlion89@gmail.com*

** Department of Computer Science, Pondicherry University, Pondicherry, India, *Email: ssivasathya@gmail.com*

in this method represents a single rule and these rules are implemented as an array of floating-point numbers. The rule consists of terms which specify the range for each attribute that is part of the rule. The two elements considered for each attribute represents the upper and lower bounds of the range for each attribute. Thus GSO members are employed to optimize the range intervals of each attributes in a multi-dimension search space and also to discover terms of each classification rule. In each iteration, the training data gets reduced by the number of records that satisfy the generated rule in the previous iteration. Stopping criterion like threshold value for maximum number of training data available in each iteration (max Uncovered Cases), number of iteration, etc., will stop the rule induction process and thereby derive the final rule set.

The implementation of the GSO algorithm for generation of classification rules from a dataset is a new and novel idea for promising research. GSO algorithm has members in the population that interact with one another to achieve an efficient, combined resultant behavior for the complete population, thereby generating an effective, strong and high-quality solution for an optimization problem having a large search space. In terms of generation of classification rules for supervised classification, GSO algorithm has the potential to generate relevant, high quality and optimized rules set based on a search strategy that is highly flexible and robust. The basic advantage of GSO over other traditional search strategies is that, it performs a global search as well a local search using a GSO member whereas the other methods are stuck in a local maxima because of their search strategies like hill-climbing search etc. Thus GSO may never or is less likely to get trapped in local maxima, in comparison to the other local search strategies [3].

In this paper, tests have been carried out on several publicly available datasets to show the efficacy of the generated rules using the GSO algorithm. The paper is presented in the following sections: Section 2 presents some of the works that are related to the method proposed in the paper. Section 3 gives the details of GSO algorithm and Section 4 is describing the rule generation strategy of GSO, GSO Miner: a supervised classification rule miner based on GSO. The experimental analysis is given in Section 5 and Section 6 concludes the proposed methodology.

2. RELATED WORK

Previously considerable amount of academic research work has been carried out in the use of ACO, PSO and GA for the purpose of generation of classification rules.

1. Ant-Miner. Parpinelli, Lopes and Freitas et.al in 2002 [4] proposed a data mining method using Ant Colony Optimization (ACO) to discover the rules for classification using datasets having only nominal attributes. They proposed Ant Colony Optimization (ACO) for generation of rules for classification and called the algorithms as Ant-Miner algorithm. The ant miner algorithm was inspired from the search strategy of ants in an ant colony and was combined with processes of information theory. Ant-Miner has operators based on heuristics measured using entropy values and the algorithm uses a sequential covering approach to generate rules of classification. Ant-Miner was designed for dataset with nominal attributes however for dataset with continuous attributes; there was a requirement of a pre-processing step which involved discretization of the continuous attributes. Later Fernando E.B. Otero et al. in 2008 [5], solved this issue for dataset with continuous attributes by implementing cAnt-Miner wherein continuous attributes are discretized using an entropy-based discretization method during the process of construction of the rules, thus avoiding the pre processing step of discretisation used in case of Ant-Miner.

2. PSO-Miner. In 2003, T. Sousa et al. [6] proposed a PSO variant Constricted Particle Swarm Optimiser (CPSO), for use in generation of classification rules. They had used CPSO since it was a more qualified version of PSO in dealing with continuous attributes and temporal complexity. The data sets used were from biology and medical science domains. Certain preprocessing tasks were carried out on the dataset in order to generate a normalized data for nominal, real and integer data. Then classification rules were generated using the CPSO algorithm and the results were compared against J48 algorithm. They were able to produce

better results by employing rule pruning and rule set cleaning, thereby generating better rule sets for classsification. In 2004, T. Sousa et al. [7] again implemented a Data Mining algorithm to discover classification rules using different three different versions of Particle Swarm Optimizer (PSO) namely CPSO, Linear Decreasing Weight Particle Swarm Optimiser (LDWPSO) and Discrete Particle Swarm Optimiser (DPSO). The PSO algorithms were compared with a rule generation algorithm based on Genetic Algorithm and a decision tree Algorithm (J48). Three nested processes were implemented in the PSO rule generation algorithm namely, the rule discovery process, the covering process and the validation process. PSO and GA forms part of the rule discovery process and is used to generate the rules for classification. The covering process uses the generated rule on the training dataset to identify the records that satisfy the particular rule set and then removing those records from the training dataset to be used for the subsequent iteration. This process is sequentially repeated to generate a set of rules for classification. The validation process determines the n fold cross validation accuracy using the generated rules on the training dataset.

In 2004, Yu Liu et al. [8] also proposed a PSO algorithm for the discovery of rules for classification. This proposed algorithm was able to work with all real-valued attribute members (both continuous and categorical data). Two benchmark data sets, Zoo and Wine data set were used to validate the results and the proposed method generated very good rule sets with each rule having a small number of conditions per rule, very few rules per rule set and also demonstrated good predictive accuracy and generalization ability. Wang, Sun and Zhang [9] in 2006 also employed PSO for generation of rules for classification. The proposed method fared comparatively much better than both Ant-Miner and ESIA (genetic algorithm classifier) on standard public domain data sets achieving greater classification accuracies and smaller rule sets. A novel evaluation function was implemented as part of generation of classification rules. Liu XiaoPing et al. [10] in 2008, proposed PSO for generation of rules for classification of a remote sensing dataset. In this research work, bands of the remote sensing dataset were used as the attributes and rules generated using certain optimized ranges of the attribute values/bands. The rules were used to classify each record or pixel in to a specific land use types. The proposed method was compared See5.0 decision tree model and was able to give a much higher classification accuracy.

3. PSO/ACO-Miner. N. Holden et al. [11] in 2005 proposed a hybrid PSO/ACO Miner algorithm for generation of rules for classification. Since PSO was not very effective with categorical attributes, ACO/ Ant-Miner was used to cope with these attributes. Ant-Miner was not very effective with continuous attributes which was solved by the use of PSO. Thus this proposed methodology was able to address the disadvantages of both ACO as well as PSO by using a hybrid model. The proposed hybrid method was compared with an existing PSO algorithm method. N. Holden et al. [12] in 2007 proposed several modifications to the hybrid PSO/ACO2 algorithm for generation of rules for classification. They had implemented two different methods to evaluate the fitness of a rule or the rule quality functions and showed the variation of performance results on the choice of rule quality functions. 16 benchmark datasets were used to validate the results of the algorithm and comparisons performed against two well known rule induction algorithms.

4. LA-Miner. In 2011, S.H. Zahiri [13] proposed a new data mining algorithm using learning automata called Learning Automata (LA-Miner). These are adaptive decision making units that perform the function of deriving the rule sets by interacting with the environment or the search space. The response of the environment pays an important role and is used by the function in order to select the rule. Classification rules have been generated using LA. LA-miner is compared with CN2, ACO, PSO and GA for performance. One of the main advantages of this method is that there is no requirement of having the knowledge of the environment in which the automaton is operating and also the analytical knowledge of the function that is required to be optimized.

5. Fuzzy-GA Rule Miner. Fuzzy methods have been used effectively to produce fuzzy classification rules for the purpose of supervised machine learning. EmelKýzýlkayaAydogan et al. (2012) [14] introduced a fuzzy rule-based classifier (FRBCS) with hybrid heuristic approach (called hGA) to solve classification

problems related to high dimensional space using fuzzy rule-based classification systems. In this study five fuzzy labels have been used per variable to define the classification rules. Triangular membership function has been used to create uniform fuzzy partitions. GA has been used for generation of the rules wherein each chromosome represented a rule. Integer-programming formulation (IPF) is used for final rule selection from the rule set.

6. GSO Algorithm. Group search optimization (GSO) is a nature inspired algorithm and has been effectively utilized for solving optimization problems. Earlier in 2006, He et al. successfully modelled GSOANN [15] classification method, in which, Group Search Optimizer is used to optimize the three layer feed forward Artificial Neural Network. GSO members are used to optimise the parameters of a three layered feed forward ANN and the weights. The performance was assessed using two real world classification problems. GSOANN provided better convergence and generalisation performances on the benchmark datasets as compared to other machine learning techniques.

7. Literature Summary. Thus from the extensive literature survey, it can be seen that classification rule discovery has effectively and efficiently improved the performance of standard machine learning approaches especially in case of evolutionary algorithms and swarm intelligence algorithms. Based on these successful inferences from the literature it was decided to evaluate the feasibility of using the GSO algorithm for the purpose of supervised classification. Based on the literature available, it has been seen that GSO has been used only for the purpose of unsupervised classification till now and GSO for generation of classification rules for supervised classification is a novel approach. GSO successfully combines global and local search performing exploration and exploitation and thereby encompassing the advantages of generality, robustness, efficiency of global search and the quality of rapid convergence toward local minima of local search. It has been seen that GSO can handle a variety of optimization problems to include large scale, high dimensional problems and real world applications.

3. GSO ALGORITHM

Group search optimization (GSO) [11] is a bio inspired optimization algorithm inspired by the search strategies of animals. GSO Optimization uses the producer–scrounger (PS) model and searches the local search space using the scanning mechanism. A group denotes the population of the GSO and the individual animals are called the GSO members. Three types of members are there in a group namely producer, scroungers and rangers. Producers are the fittest individual of the group and they search the local search space by way of animal scanning mechanism; scroungers move towards the producers in each iteration and rangers perform global search by way of random walks.

In an n-dimensional search space, the ith member at the kth searching bout (iteration) has a position, $X_i^K \in \mathbb{R}^n$ and a head angle $\Phi_i^k = \left(\Phi_{i1}^k, \Phi_{i2}^k, \ldots \ldots \Phi_{i(n-1)}^k\right) \in \mathbb{R}^{n-1}$. The search direction associated with the ith member, which is represented as a unit vector $D_i^k\left(\Phi_i^k\right) = \left(d_{i1}^k, d_{i2}^k, \ldots \ldots d_{in}^k\right) \in \mathbb{R}^n$ as presented in equation(1).

$$\begin{cases} d_{i_1}^k &= \prod_{q=1}^{n-1} \cos\left(\Phi_{i_q}^k\right) \\ d_{i_f}^k &= \sin\left(\Phi_{i(f-1)}^k\right).\prod_{q=j}^{n-1} \cos\left(\Phi_{i_q}^k\right) (j = 2,\ldots m-1) \\ & d_{i_n}^k = \sin\left(\Phi_{i(n-1)}^k\right) \end{cases} \quad (1)$$

The scanning mechanism of each member is an n-dimensional space that has a maximum pursuit angle of $\Theta_{\max} \in \mathbb{R}^1$ and maximum pursuit distance of $l_{\max} \in \mathbb{R}^1$. The producer X_p^k scans at zero degree and laterally

by randomly sampling three points in the search space to find the best position at each iteration. If the producer cannot find a better search position after '*a*' iterations, it will turn its head back to zero degree. Scroungers follow the producer using the equation 2.

$$X_i^{k+1} = X_i^k + r\ o(X_p^k - X_i^k) \tag{2}$$

where $r \in \mathbb{R}^n$ is a uniform random sequence in the range (0,1). Operator "*o*" is the Hadamard product or the Schur product, which calculates the entry wise product of the two vectors. If any of the GSO member finds a better position than the current producer, then that GSO member becomes the producer in the next iteration. Rangers move to the new point based on a random head angle and a random distance using the equation 3:

$$\begin{cases} l_i = a.r_1.l_{\max} \\ X_i^{k+1} = X_i^k + l_i.d_i^k.(\Phi^{k+1}) \\ \Phi^{k+1} = \Phi^k + r_2\alpha_{\max} \end{cases} \tag{3}$$

Here, $r_1 \in \mathbb{R}^1$ is a normally distributed number with mean 0 and standard deviation 1 and $r_2 \in \mathbb{R}^{n-1}$ is a uniformly distributed random sequence in the range (0, 1).

4. GSOMINER–THE PROPOSED METHODOLOGY FOR CLASSIFICATION RULE DISCOVERY

This section discusses in detail the proposed GSOMINER in greater detail and it is divided into following parts: GSOMINER Overview, rule construction, rule pruning and classification given in the experimental results.

1. GSOMINER Overview. The antecedent of a generated rule consists of a set of terms given by an attribute value pair. A term is a triple <attribute, operator, value >, where the value is determined by range of the attribute. The operator used in this implementation is "=" in the case of categorical/nominal attributes and "≤" or "≥" in the case of continuous attributes. The rule format is as given:

IF *attr*1 ≤ *Value* _1 AND *attr*1 ≥ *Value* _ 2 AND *attr 2* = *Value* _ 3 AND …..

attr j ≤ *Value* _4 AND *attr j* ≥ *Value* _ 5 THEN *Class* _ *x*

Each individual GSO member encodes a single rule and the rules are derived as an array of floating-point numbers and each attribute is represented by two elements on this array. Real and Integer attributes are assigned with a lower bound and an upper bound in the array in order to implement a value range. However in case of nominal attributes having more than one value, a different approach has to be taken. Conversion of the nominal attribute into a binary array would increase computational cost and complexity. For instance, consider the nominal attribute 'colour' having 5 values: blue, black, red, orange and white. This attribute value can thus have possible 5 binary values ("true" or "false" for each original nominal

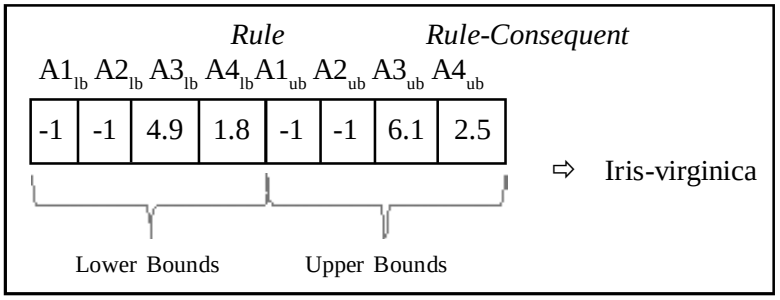

Figure 1: A sample rule represented by a GSO member

value) but as a consequence this will increase the number of attributes (and thereby dimensionality of the search space). Alternatively, the another less complex method is to have the original nominal values be converted into sequential numbers, i.e; "1, 2, 3, 4, 5". Figure 1 shows a typical rule represented by an individual member in GSOMiner method.

The above rule signifies,

If A3 $\geq$ 4.9 *AND* A3 $\leq$ 6.1 *AND* A4 1.8 *AND* A4 2.5 *THEN Class* = 'Iris-virginica'

In this case, the upper and lower bound values with -1 are ignored. In the proposed method, only values within a range are considered corresponding to each attribute. Therefore, if any of the lower or upper bounds provides a value -1 for an attribute then both bounds are replaced by -1.

The GSO algorithm discovers the rules and returns the best quality rule. The proposed method sequentially discovers one classification rule at each GSO iteration. This rule is then used to remove all those records or instances that satisfy this rule. Then the next iteration is run and GSO algorithm iteratively builds a sequential rule set. There are certain stopping criterion defined by the user like the number of iteration, max unclassified instances etc. A default rule is defined that correctly classifies instances not classified by the previous rules and this default rule is then added to the rule set. The default rule is generated using the max class instances left in the remaining instances.

2. Rule Generation. The rule set (RS) is initialized with an empty set. Then, for each class the algorithm performs a WHILE loop and the rules are created. The best discovered rule (Rule) for the iteration is identified, predicting the class with maximum covered instances of that class. Rules are pruned in a rule pruning step, to remove irrelevant terms. There is a parameter named '*No_of_members'* in the algorithm, which determines the number of members involved in the rule construction procedure. After the final pruning, the records that satisfy the discovered best rule are removed from the training set. The WHILE loop is performed until the stopping criterion is reached i.e either the no of iterations are reached or as long as the training set reaches the number of maximum uncovered cases (max Uncovered Cases).

Each rule has terms which correspond to specific zones of the range of the attribute. The range of each attribute is determined from the maximum and minimum value of the readings in the attribute. Specific zones (between certain upper value and lower value) are selected by each GSO member in an attribute as part of the rule construction process as shown in figure 2. The range that is to be optimized has two dimensions per attribute, upper bound (*ub*) and lower bound (*lb*) of the range. The range is represented in terms of the rule conditions for each GSO member. If the two bounds crosses over, (i.e., $xlb_0 > xub_0$) then the values of the terms of the bounds are interchanged with each other and If the values of terms of two bounds are equal, (i.e., $xlb_0 = xub_0$) then only one term is considered with an "=" operator and the other term is automatically omitted from the generated rule. Suppose there are 'm' GSO members and 'n' attributes, then the position of the i^{th} GSO member is represented as (xlb_{i1}, xub_{i1}, xlb_{i2}, xub_{i2},……, xlb_{in}, xub_{in}). Initially, the lowest and highest values of each attribute are found from the training set instances. Then, each GSO member's initial position (for lower and upper bound) is set to a randomly chosen value from the attribute values in the training set. If >, they will be reciprocally replaced. After each iteration run, the fitness value for each GSO member is calculated, and the member with maximum fitness value is denoted as the producer. After the stopping criterion of maximum iteration or the convergence test is reached, the rule associated with the producer is added to the rule set. Figure 3 represents the pseudo-code for the proposed GSO-Miner method.

3. Rule Quality Measurement. A classification rule is discovered after each GSO iteration. It is necessary to estimate the quality of every candidate rule. A measure is required to be used in the training phase to estimate the performance of the rule in the testing phase. This measure will help to optimize a rule's quality based on a fitness function. In the proposed work, the quality measure is given in equation (4).

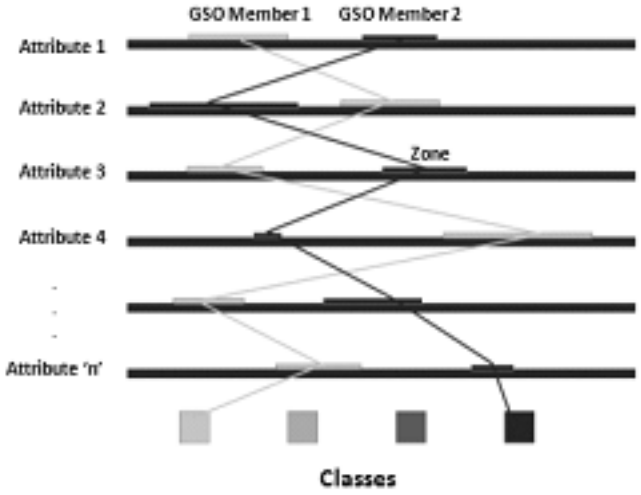

Figure 2: The principle of rule generation by GSOMiner

```
Input Training Set TS = Φ;
Input Rule-Set RS = Φ;

WHILE (Training Set Length >maxUncoveredCases)
    K=0;
INITIALIZE population with GSO members
    WHILE (k<max_No_of_Iteration AND
    Convergence_Test is not met)
    FOR i = 1 to GSO_Population_Size
    {
            Update Data Instances(TS);
            Determine Rule Consequent of i^th GSO member;
            Calculate the Rule Quality
        associated with i^th GSO member;
            Prune i^th GSO member Rule;
    }
    Assign producer index to max fit member;
    Randomly select scroungers(80% members);
    Select rangers(rest members);
    Set k=k+1;
    END WHILE
BestRule =Finally Discovered Rule Associated With Producer;
RS=RS ∪BestRule
TS=TS – { Training Instances covered By the BestRule}
END WHILE
```

Figure 3: Pseudocode for GSOMiner method

$$Q(x) = \begin{cases} \dfrac{TP}{(TP+PN)} \times \dfrac{TN}{(TP+PP)} & \text{if no of training examples} > \text{minlIncovered Cases Per Rule} \\ \dfrac{TP}{(TP+PN)} & \text{if all training example are in same class} \\ 0.0 & \text{if no. of covered examples} < \text{min Covered Cases Per Rule} \end{cases} \quad (4)$$

where TP, FN, FP and TN are, respectively, the number of true positives, false negatives, false positives, and true negatives associated with the rule.

- **True Positives** (TP): Records that match the rule. These are the correct predictions.
- **False Positives** (FP): Records that match the rule conditions however the class prediction is not correct. These are the incorrect predictions.
- **False Negatives** (FN): Records that do not match the rule conditions however belong to correctly classified class. These are the uncovered cases.
- **True Negatives** (TN): Records that do not match the rule.

The rule having maximum fitness or quality is considered as the best discovered rule and added to the rule-set after each iteration.

4. Rule Pruning. The generated rules need to be pruned in order to optimise the generated rules and also to make the rule set more comprehensible. The unnecessary attribute_value terms are removed from the rules in this process of rule pruning. This is done by iteratively removing each attribute term and the process is stopped, whenever the newly obtained rule has the same or higher quality value than the original rule.

5. EXPERIMENTAL ANALYSIS

The proposed methodology is implemented in Java with the help of the open source java packages of WEKA 3.7.9 machine learning tool developed by Waikato University [18]. Performance is evaluated on the UCI benchmark data sets. The datasets are available at the University of California at Irvine (UCI) Machine Learning repository database [19].

1. Experimental Settings. The dimension of each GSO member is given by $d = 2n$; where n is the number of attributes in the dataset. The initial position of each GSO member is generated uniformly at random i.e the values for the terms of both upper and lower bounds, from values of the attributes of a randomly chosen record from the training set. The head angle Φ^0 for each GSO member is initially set to π□/4. The value of constant 'a' is derived from the function round ($\sqrt{d+1}$) where d is the dimension of the search space or the number of classes in the training dataset. Maximum pursuit angle for each GSO member is given by Θ_{max} and the value taken in this paper is $\pi/_a{}^2$. The maximum turning angle for each GSO member is Θ_{max} and the value is given by $\Theta_{max}/2$. The maximum pursuit distance for each GSO member is l_{max} and calculated using equation (5) given below:

$$l_{\max} = \|U - L\| = \sum_{i=1}^{m} (U_i - L_i)^z \tag{5}$$

In the proposed method $U_i = X_{ub}$ and $L_i = X_{lb}$, has been discussed earlier. In each generation, one best fit member is treated as the producer, 20% of the total GSO population are chosen randomly as rangers and remaining members will perform scrounging. Table 1 gives the values of each parameter for GSOMiner algorithm:

Table 1
Algorithmic Parameters

Number of GSO members	30
Folds for Cross Validation	10
Minimum Covered Examples per rule	05
Maximum number of Iterations	100
Maximum number of uncovered cases	10

2. Performance Evaluations. Table.2 presents the generated rule-set details along with the classification accuracies given by GSOMiner on different UCI datasets.

Table 2
Generated Rule-Set details and predicted accuracy of GSOMiner approach

DataSet	*Rule-Set Details*		*Accuracy (%)*
	Rules	*Conditions per Rule*	
Australian	7.1 ± 0.23	23.8 ± 2.01	85.8 ± 1.33
Breast-w	6.2 ± 0.13	22.4 ± 0.88	95.14 ± 0.83
Breast cancer	6.3 ± 0.15	26.2 ± 1.72	73.7 ± 3.15
Crx	7.4 ± 0.16	23.2 ± 1	86.09 ± 1.9
Diabetes	7.4 ± 0.27	20.8 ± 1.87	76.29 ± 1.47
Ecoli	8.9 ± 0.41	26.4 ± 1.81	73.02 ± 3.11
Glass	10 ± 0.61	33.6 ± 3.19	60.35 ± 3.25
Heart-c	5.7 ± 0.15	18.2 ± 1.05	77.42 ± 2.77
Heart-statlog	5.8 ± 0.2	17.4 ± 1.55	79.26 ± 3.08
Hepatitis	5 ± 0.21	20.8 ± 2	79.75 ± 3.36
Ionosphere	5.7 ± 0.26	17.4 ± 1.03	90.93 ± 1.32
Iris	4.4 ± 0.16	8.2 ± 0.81	95.33 ± 1.02
Segment	13.2 ± 0.55	46.8 ± 3.27	76.02 ± 1.97
Sonar	5.3 ± 0.21	13.8 ± 1.28	68.52 ± 5.53
Tic-tac-toe	8.8 ± 0.13	35.6 ± 1.81	71.82 ± 1.36
Vote	4 ± 0	7.4 ± 0.31	95.2 ± 0.85
Wine	5.6 ± 0.22	12 ± 1.03	78.47 ± 3.28

The classification accuracies achieved using GSOMiner on the standard UCI datasets have been compared with the results achieved using different versions of Ant-Miner. The results of GSOMiner on the datasets used in the performance of different versions of Ant-Miner in research work by Otero and Alex A. Freitas et al. in 2009 [16] is given in Table 3 and the graphical representation of these results are shown in Figure 4.

Table 3
Comparisons on classification accuracies achieved using GSOMiner and different versions of Ant-Miner

Data-Set	*cAnt-Miner PB*	*cAnt-Miner PB [E]*	*cAnt-Miner PB [D]*	*cAnt-Miner PB[D+E]*	*GSOMiner*
Breast-w	94.29	94.34	94.09	94.60	95.14
Breast cancer	72.32	75.27	70.59	73.77	73.7
Glass	73.94	73.11	72.73	73.52	60.35
Heart-c	55.50	55.21	54.57	54.83	77.42
Hepatitis	78.78	78.55	79.50	78.83	79.75
Ionosphere	89.65	89.95	89.32	90.58	90.93
Iris	93.24	93.13	94.33	94.47	95.33
Wine	93.57	94.51	94.18	95.0	78.47

The results of GSOMiner as compared with the various versions of Ant-Miner and other algorithms implemented by Freitaset. al in 2012 [17] is given in Table 4 and graphically represented shown in Figure 5. These comparisons demonstrate that GSOMiner fares very well in comparison to these Ant-Miner variations on most of the datasets.

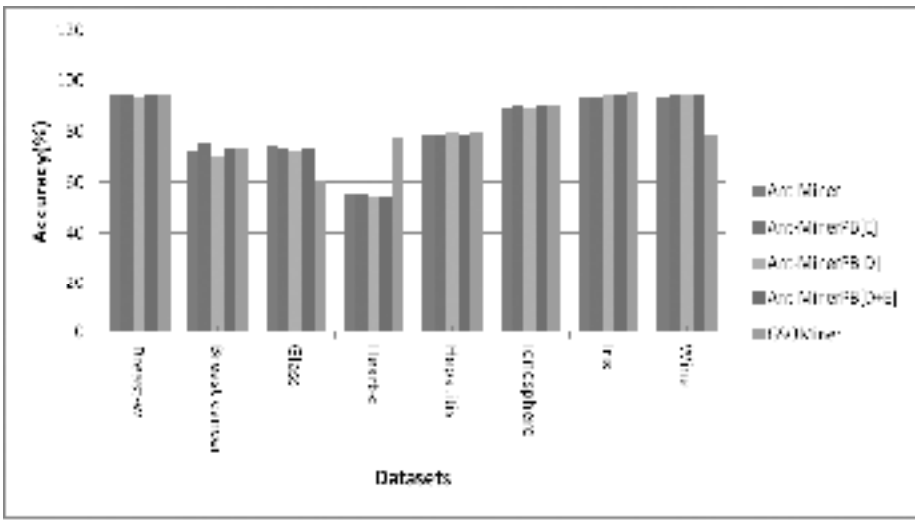

Figure 4: Comparison of GSOMiner rule Classifier with different versions of CAnt-Miner

Table 4
Comparisons on classification accuracies achieved using J48, different versions of Ant-Miner and GSOMiner

Data-Set	J48	Ant-Miner	cAnt-Miner	cAnt-Miner-MDL	cAnt-Miner2	GSOMiner
Australian	85.07	83.41	84.87	84.51	84.48	85.8
Breast-w	92.63	90.39	93.88	93.30	93.94	95.14
Crx	85.29	83.21	85.12	85.11	85.79	86.09
Glass	62.49	48.08	64.60	62.85	66.72	60.35
Heart	78.15	74.77	74.67	75.67	76.89	79.26
Hepatitis	82.15	70.17	84.92	84.74	84.74	79.75
Ionosphere	88.57	88.45	86.60	87.17	86.14	90.93
Wine	93.30	83.23	89.25	88.73	92.27	78.47

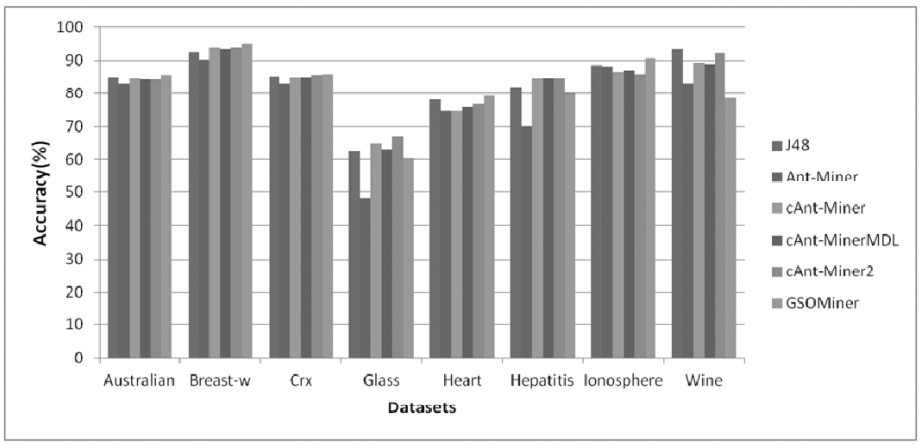

Figure 5: Comparison of GSOMiner rule Classifier with different versions of CAnt-Miner and J48 algorithms

The GSOMiner algorithm is also compared with the results as obtained using PSO/ACO2 algorithm as given in N. Holden et al. [12]. Table 5 gives the comparison of the classification accuracy results achieved using standard UCI datasets and the visual representation of the comparative accuracy outcomes for PSO/ ACO2 and GSOMiner on the UCI datasets are given by Figure 6.

Table 5
Predicted accuracy details for cAnt Miner, PSO/ACO2 and GSOMiner

Data-Set	PSO/ACO2	GSOMiner
Australian	84.93 ± 3.73	85.8 ± 1.33
Breast-w	95.91 ± 2.51	95.14 ± 0.83
Breast cancer	72.62 ± 6.84	73.7 ± 3.15
Crx	85.6 ± 2.84	86.09 ± 1.9
Diabetes	72.67 ± 4.98	76.29 ± 1.47
Ecoli	82.15 ± 3.69	73.02 ± 3.11
Glass	70.95 ± 7.5	60.35 ± 3.25
Heart-c	77.38 ± 5.45	77.42 ± 2.77
Heart-statlog	81.11 ± 6.16	79.26 ± 3.08
Hepatitis	78.75 ± 13.75	79.75 ± 3.36
Ionosphere	88.06 ± 4.91	90.93 ± 1.32
Iris	94.67 ± 5.26	95.33 ± 1.02
Segment	96.67 ± 1.17	76.02 ± 1.97
Sonar	75.05 ± 9.11	68.52 ± 5.53
Tic-tac-toe	100.0 ± 0.0	71.82 ± 1.36
Vote	97.03 ± 3.28	95.2 ± 0.85
Wine	89.87 ± 4.96	78.47 ± 3.28

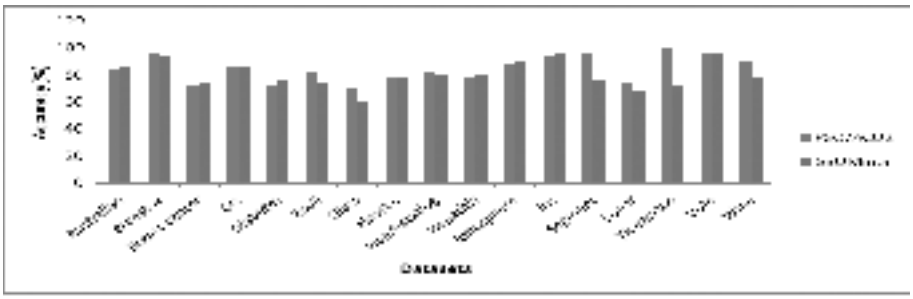

Figure 6: Comparison of PSO/ACO2 and GSOMiner algorithms

6. CONCLUSION

In this paper a GSO is used for generation of rules for classification of a dataset. The experiments show that this real encoded GSO being used as a GSOMiner is effective and efficient in performing supervised classification tasks. The rules discovered by GSOMiner for all the standard datasets have been compared with other similar algorithms and found to be generally with high accuracy, generalization and comprehensibility. The results show that this approach has good performance for rule discovery on continuous data. Each rule in GSO-Miner is optimized individually, without interacting with the quality of other rules. The future work resides on improving the performance of proposed algorithms in terms of time complexity and applying them to the real world classification problems such as medical datasets, satellite images etc. More modifications t this approach of using GSOMiner or use of an hybrid technique may help in generation of better results..

References

[1] Fidelis, Marcos Vinicius, H. S. Lopes, and A. A. Freitas. “Discovering comprehensible classification rules with a genetic algorithm.” Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. Vol. 1. IEEE, 2000.

[2] Liu, Bo, Hussein A. Abbass, and Bob McKay. "Classification rule discovery with ant colony optimization." Intelligent Agent Technology, IEEE/WIC/ACM International Conference on. IEEE Computer Society, 2003.

[3] S. He, Q. H. Wu, and J. R. Saunders, "Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior," IEEE Transactions On Evolutionary Computation, vol. 13, no. 5, pp. 973-990, October 2009.

[4] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., vol. 6, no. 4, pp. 321–332, 2002.

[5] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "c Ant-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes," pp. 48–59, 2008.

[6] Sousa, Tiago, Arlindo Silva, and Ana Neves. "Particle swarm based data mining algorithms for classification tasks." Parallel Computing 30.5 (2004): 767-783.

[7] Sousa, Tiago, Arlindo Silva, and Ana Neves. "A particle swarm data miner." Progress in Artificial Intelligence. Springer Berlin Heidelberg, 2003. 43-53.

[8] Y. Liu, Z. Qin, Z. Shi, and J. Chen, "Rule Discovery with Particle Swarm Optimization," AWCC 2004, LNCS 3309, pp. 291–296, 2004.

[9] Wang, Ziqiang, Xia Sun, and Dexian Zhang. "Classification rule mining based on particle swarm optimization." Rough Sets and Knowledge Technology. Springer Berlin Heidelberg, 2006. 436-441.

[10] Liu XiaoPing, LI Xia, PENG XiaoJuan, LI HaiBo and HE JinQiang, "Swarm intelligence for classification of remote sensing data," Sci China Ser D-Earth Sci., vol. 51, no. 1, pp. 79-87, Jan. 2008.

[11] Holden, Nicholas, and Alex A. Freitas. "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data." Proceedings of the IEEE swarm intelligence symposium (SIS). 2005.

[12] N. Holden and A. A. Frietas, "A Hybrid PSO / ACO Algorithm for Classification,"GECCO'07, July 7–11, 2007, London, England, United Kingdom. 2007.

[13] S.-H. Zahiri, "Classification rule discovery using learning automata," Int. J. Mach. Learn. Cybern., vol. 3, no. 3, pp. 205–213, Nov. 2011.

[14] E. K. Aydogan, I. Karaoglan, and P. M. Pardalos, "hGA: Hybrid genetic algorithm in fuzzy rule-based classification systems for high-dimensional problems," Appl. Soft Comput., vol. 12, no. 2, pp. 800–806, Feb. 2012.

[15] S. He, Q.H. Wu and J.R. Saunders, "A Group Search Optimizer for Neural Network Training," Computational Science and Its Applications-ICCSA 2006 Lecture Notes in Computer Science Volume 3982, 2006, pp 934-943.

[16] Otero, Fernando EB, Alex AlvesFreitas, and Colin G. Johnson. "Handling continuous attributes in ant colony classification algorithms." Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on. IEEE, 2009.

[17] Medland, Matthew, Fernando EB Otero, and Alex A. Freitas. "Improving the cAnt-MinerPB Classification Algorithm." Swarm Intelligence. Springer Berlin Heidelberg, 2012. 73-84.

[18] "WEKA software, Machine Learning, The University of Waikato, Hamilton, New Zealand." Available: http://www.cs.waikato.ac.nz/ml/weka/.

[19] "UCI Machine Learning Repository." Available: http://mlearn.ics.uci.edu/MLRepository.html.