


Ant Miner: A Hybrid Pittsburgh Style Classification Rule Mining Algorithm

Bijaya Kumar Nanda, BIET, Odisha, India

Satchidananda Dehuri, Fakir Mohan University, Balasore, India

 <https://orcid.org/0000-0003-1435-4531>

ABSTRACT

In data mining the task of extracting classification rules from large data is an important task and is gaining considerable attention. This article presents a novel ant miner for classification rule mining. The ant miner is inspired by researches on the behaviour of real ant colonies, simulated annealing, and some data mining concepts as well as principles. This paper presents a Pittsburgh style approach for single objective classification rule mining. The algorithm is tested on a few benchmark datasets drawn from UCI repository. The experimental outcomes confirm that ant miner-HPB (Hybrid Pittsburgh Style Classification) is significantly better than ant-miner-PB (Pittsburgh Style Classification).

KEYWORDS

Ant Colony Optimization, Ant Miner, Ant Miner-HPB, Ant Miner-PB, Classification Rule Mining, Data Mining, Pheromone, Simulated Annealing

1. INTRODUCTION

Learning classification rules from instances have been a subject of intense study in data mining (Han & Kamber, 2006; Dehuri, Ghosh & Ghosh, 2008; Panda, Dehuri, & Patra, 2015). Based on various theories and techniques, many different algorithms have been proposed to generate classification rules (Dehuri & Mall, 2006; Dehuri, et al., 2008, & Kalia, et al., 2018). However, there are three common and important factors for classification rule learning: higher predictive accuracy, smaller rule sets, and shorter running time (Freitas, 2002;). More precisely, higher predictive accuracy means more efficient application; smaller rule sets enable better understanding for the user, and shorter running time means that the algorithm can be applied to online systems and address the scalability issue (Kalia, et al., 2018).

Ant miner is an application for extracting classification rules from data by simulating the behaviours of real ant colonies (Parpinelli, Lopes, & Freitas, 2001). Aiming at the insufficiencies of ant-miner, researchers have proposed some improvement strategies resulting new versions

DOI: 10.4018/IJAIML.2020010104

This article, originally published under IGI Global's copyright on December 6, 2019 will proceed with publication as an Open Access article starting on January 18, 2021 in the gold Open Access journal, International Journal of Artificial Intelligence and Machine Learning (converted to gold Open Access January 1, 2021), and will be distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

of ant-miner including new heuristic function formula, a new pheromone updating method, and a state transition rule, which can get higher accuracy rate (Otero, Freitas & Johnson, 2008; Liu, Abbass, & McKay, 2004; Parpinelli, Lopes, & Freitas, 2002). A comparative analysis and survey of ant colony optimization based rule miner can be obtained in (Ali & Shahzad, 2017). On the other hand, in rule-based classifier, many objectives are involved and regain simultaneous optimization, for example, minimization of the number of rules or length of rule, maximization of classification accuracy of the rule, maximization of interestingness, etc (Dehuri, Ghosh, & Ghosh, 2008). In lieu of these criteria, classification rule mining problem is not restricted as single objective problem. It is also an attractive field of research. For many objective optimization researches, although we realize this classification rule mining is a multi-objective optimization problem but by giving higher priority to the objective of classification accuracy, a novel algorithm has been developed by us. We explore the search space by giving proper attention to both good and bad solution with certain probabilities (this concept is derived from simulated annealing (Aarts & Korst, 1989; Laarhoven & Aarts, 1987)). However, overall rule discovery process is motivated by ant colony optimization method (Mahapatra & Patnaik, 2018; Angus & Woodward, 2009). The combined approach of both stochastic approaches drives us to uncover hidden pattern very effectively.

2. PRELIMINARIES

In this section, we discuss two basic algorithmic paradigms like ant colony optimization and simulated annealing in following Subsections.

3. ANT COLONY OPTIMIZATION

Since the early 1990's, several collective behavior (like social insects and bird flocking) inspired algorithms have been proposed and applied in many optimization problems with inherent intractability (Abraham, Grosan, & Ramos, 2006; Bonabeau, Dorigo, & Theraulaz, 1999; Tao, 2018). Ant Colony Optimization (ACO) (Blum, 2005; Dorigo & Stutzle, 2006; Dorigo, Maniezzo, & Coloni, 1996; Bonabeau, Dorigo & Theraulaz, 1999) is one of the most popular algorithms of them and was introduced around 1990. Ants are social insects, living in colonies and exhibit an effective collective behavior. Although each ant is relatively a simple insect with limited individual abilities, a swarm of ants has the ability to find the shortest path from their nest to food. Further, it was discovered that most of the communication amongst individual ants is based on the use of a chemical, called pheromone that is dropped on the ground. As ants walk from a food source to the nest, pheromone is deposited on the ground, creating in this way a pheromone trail on the path used. Shorter paths will be traversed faster and by consequence, will have stronger pheromone concentration than longer paths over a given period of time. The more pheromone path contains, the more attractive it becomes to be followed by other ants. Hence, as time goes by, more and more ants will prefer the shorter path, which will have more and more pheromone. At the end, almost all ants will be following a single path which usually will represent the shortest path between the food source and the nest.

An ant probabilistically chooses a path to follow based on heuristic information and the amount of pheromone deposited by previous ants. The inter-active process of building candidate solutions and updating pheromone values allows an ACO algorithm to converge to optimal or near optimal solutions (Liu, Zhang, & Yu, 2019; Al-Behadili, 2018). Algorithm 1 presents a high-level pseudo code of a basic ACO procedure, comprising four main steps: Initialization (), Construct-Ant-Solution (), Local-Search (), and Update-Pheromone ().

Algorithm 1. High-level pseudo code of a basic ACO algorithm

```

BEGIN
INITIALIZATION ();
WHILE (Termination condition not met) DO
BEGIN
Construct-Ant-Solutions ();
        Apply Local-Search (); // optional
Update-Pheromones ();
END
END WHILE
RETURN Best Solution;
END
    
```

3.1. Construction of Ant Solution ():

Candidate solutions are created by simulating the movement of artificial ants on the construction graph. Each ant incrementally creates a candidate solution by moving through neighbor vertices of the construction graph G . Hence, a candidate solution is represented by the list of visited vertices, which corresponds to a path in the construction graph. The vertices to be visited are chosen in a stochastic decision process, where the probability of choosing a particular neighbor vertex depends on both the problem dependent heuristic information η and the amount of pheromone τ associated with the neighbor vertex (η_j and τ_j , respectively) or the edge leading to the neighbor vertex (η_{ij} and τ_{ij} , respectively). Given an ant currently located at vertex v_i , the probability of selecting a neighbor vertex v_j is given by:

$$P_{ij} = \frac{[\tau_j]^\alpha \times [\eta_j]^\beta}{\sum_{j=1}^{|\mathcal{F}_i|} [\tau_j]^\alpha \times [\eta_j]^\beta}, \forall j \in \mathcal{F}_i \quad (1)$$

where τ_j and η_j are the pheromone value and heuristic information associated with the j^{th} vertex, respectively, \mathcal{F}_i is the feasible neighborhood of the ant located at vertex v_i (the set of vertices that the ant can visit from vertex v_i), α and β are (user defined) parameters used to control the influence of the pheromone and heuristic information, respectively.

4. UPDATE PHEROMONE

After building the candidate solutions of iteration, the updating of pheromone trails in the construction of graph is usually accomplished in two steps, namely reinforcement and evaporation. The reinforcement step consists of increasing the amount of pheromone of every vertex (or edge, in the case that pheromone is associated with edges of the construction of graph) used in a candidates solution and it is usually only applied to the best candidate solution according to a problem dependent quality measure Q of the current iteration. Assuming that pheromone values are associated with vertices of the construction graph, a simple reinforcement rule is given by:

$$\tau_i = \tau_i + \Delta Q(CS), \forall i \in CS \quad (2)$$

where ΔQ (CS) is the amount of pheromone proportional to the quality of the candidate solution CS to be deposited. τ_i is the pheromone value associated with the i^{th} vertex of the candidate solution CS. The evaporation step consists of lowering the pheromone value of every vertex or edge simulating the natural phenomenon of pheromone evaporation in order to avoid a quick convergence of all ants toward a sub-optimal solution. Assuming that pheromone values are associated with vertices, a simple evaporation rule is given by:

$$\tau_i = (1 - \rho) \cdot \tau_i, \forall i \in G \quad (3)$$

where $\rho \in [0,1]$ is a parameter representing the evaporation factor, τ_i is the pheromone value associated with the i^{th} vertex of the construction graph G.

5. SIMULATED ANNEALING

The simulated annealing (Aarts & Korst, 1989; Hwang, 1988) procedure simulates this process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. The cooling phenomenon is simulated by controlling a temperature-like parameter introduced with the concept of the Boltzmann probability distribution. According to the Boltzmann ΔE probability distribution, a system in thermal equilibrium at a temperature T has its energy distributed probabilistically according to $P(E) = \exp\left(-\frac{\Delta E}{kT}\right)$, where k is the Boltzmann constant. This expression suggests that the system at a high temperature has almost uniform probability of being at any energy state, but at a low temperature it has a small probability of being at a high-energy state. Therefore, by controlling the temperature T and assuming that the search process follows the Boltzmann probability distribution, the convergence of an algorithm can be controlled. Metropolis, et al. in 1953 (Laarhoven & Aarts, 1987) suggested a way to implement the Boltzmann probability distribution in simulated thermodynamic systems. The same can also be used in the function minimization context.

Let us say, at any instance the current point is $x^{(t)}$ and the value at that point is $E(t) = f(x^{(t)})$.

Using the metropolis algorithm, we can say that the probability of the next point being at $x^{(t+1)}$ depends on the difference in the function values at these two points or on $\Delta E = E(t+1) - E(t)$ and is calculated using the Boltzmann probability distribution:

$$P(E(t+1)) = \min\left[1, \exp\left(-\frac{\Delta E}{kT}\right)\right] \quad (4)$$

If $\Delta E \leq 0$, this probability is one and the point $x^{(t+1)}$ is always accepted. In the function minimization context, this makes sense because if the function value at $x^{(t+1)}$ is better than that at $x^{(t)}$, the point $x^{(t+1)}$ must be accepted. The interesting situation happens when, $\Delta E > 0$ which implies that the function value at $x^{(t+1)}$ is worse than that at $x^{(t)}$. According to many traditional algorithms the point $x^{(t+1)}$ must not be chosen in this situation. But according to the Metropolis algorithm, there is some finite probability of selecting the point $x^{(t+1)}$ even though it is worse than the point $x^{(t)}$. However, this probability is not the same in all situations. This probability depends on relative magnitude of ΔE & T values. If the parameter T is large, this probability is more or

less high for points with largely disparate function values. Thus, any point is almost acceptable for a large value of T . On the other hand, if the parameter T is small, this probability of accepting an arbitrary point is small. Thus, for small values of T , the points with only small deviation in function value are accepted.

Simulated annealing is a point-by-point method. The algorithm begins with an initial point and a high temperature T . A second point is created at random in the vicinity of the initial point and the difference in the function values (ΔE) at these two points is calculated. If the second point has a smaller function value, the point is accepted: otherwise the point is accepted with a probability $\exp\left(-\frac{\Delta E}{T}\right)$. This completes an iteration of the simulated annealing procedure. In the next generation, another point is created at random in the neighborhood of the current point and the Metropolis algorithm is used to accept and reject the point. In order to simulate the thermal equilibrium at every temperature, a number of points (n) are usually tested at a particular temperature, before reducing the temperature. The algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in function values is found.

Algorithm 2. Canonical algorithm of simulated annealing

1. Choose an initial point $x^{(0)}$, a termination criterion ε .
Set T a sufficiently high value, number of iteration to be performed at a particular temperature n , and set $t=0$.
2. Calculate a neighboring point $x^{(t-1)} = N\left(x^{(t)}\right)$, Usually, a random point in the neighborhood is created.
3. If $\Delta E = E\left(x^{(t-1)}\right) - E\left(x^{(t)}\right) < 0$, set $t = t + 1$
Else
Create A random number (r) in the range $(0,1)$.
$$\text{If } r \leq \exp\left(-\frac{\Delta E}{T}\right) \text{ set } t = t + 1$$

Else go to step 2.
4. If $\left|x^{(t-1)} - x^{(t)}\right| < \varepsilon$ and T is small, terminate
else if $(t \bmod n) = 0$ then lower T according to cooling schedule .
go to step2;
Else go to step 2.

The initial temperature (T) and the number of iteration (n) performed at a particular temperature are two important parameters which governs successful working of the simulated annealing procedure. If a large initial T is chosen, it takes a number of iterations for convergence. On the other hand, if a small initial T is chosen the search is not adequate to thoroughly investigate the search space before converging to the true optimum. A large value of n is recommended in order to achieve quasi-equilibrium state at each temperature, but the computation time is more. Unfortunately, there are no unique values of the initial temperature and that work for every problem. However, an estimate of the initial temperature can be obtained by calculating the average of the function values at a number of random points in the search space. A suitable value of n can be chosen (usually between 20 to 100) depending on the available computing resource and the solution time. Nevertheless, the choice of initial temperature and subsequent cooling schedule still remain an art and usually require some trial-and-error efforts.

6. ANT MINER-PB (ANT MINER- PITTSBURGH STYLE)

In the Michigan style approach, each ant corresponds to a rule and a set of rules is represented by the colony of ants, using some mechanism to ensure that different rules cover different regions of the data space. Hence, a single run of a procedure following a Michigan approach discovers a complete list of rules. Similarly, in the Pittsburgh approach, each run of the procedure discovers a complete list of rules (the best list of rules produced over all iterations). One of the main differences between IRL/ Michigan and Pittsburgh approaches is that in the latter a complete list of rules, which constitutes an ant, is evaluated instead of a single rule, in order to guide the discovery process. As discussed in (Freitas, 2002) evaluating the quality of a rule individually, instead of the quality of a list of rules as a whole, has difficulty with the problem of rule interaction i.e., the list of best rules is not necessarily the best list of rules (Otero et al., 2012).

The classification rules discovered through Pittsburgh approach based on the ant miner is presented in the following algorithm. This algorithm to some extent overcomes the rule interaction problem of ant miner-MC (Michigan Approach).

Algorithm 3. Ant miner algorithm of Pittsburgh style (Ant Miner-PB)

```
Procedure_Ant_Miner_Pittsburgh_Style()
Input : TrainingSet;
Output : List_of_Rules ;
ComputationalSteps :
  INTIALIZATION_OF_PHEROMONES() ;
  GB_LIST  $\leftarrow$   $\emptyset$ ;
  t  $\leftarrow$  0;
  WHILE(t < maximum_IterationAndNot_Stagnation)DO
  IB_LIST  $\leftarrow$   $\emptyset$ ;
  Fori  $\leftarrow$  1 toSize(COLONY)DO
  M  $\leftarrow$  size(trainingSet);
  i_LIST  $\leftarrow$   $\emptyset$ ;
  WHILE(M > Maximum_uncovered)DO
  COMPUTATION_OF_HEURISTIC_INFORMATION(TrainingSet) ;
  RULE  $\leftarrow$  CREATERULE(TrainingSet);
  PRUNE(RULE);
  TrainingSet  $\leftarrow$  TrainingSet - Covered(Rule, TrainingSet);
  i_LIST  $\leftarrow$  i_LIST + RULE;
  ENDWHILE
  IF(Predictive_Accuracy(i_LIST)) > (Predictive_Accuracy(IB_LIST))THEN
  IB_LIST  $\leftarrow$  i_LIST;
  ENDFOR
  Update_Pheromones(IB_RULE);
  IF(Predictive_Accuracy(IB_LIST)) > (Predictive_Accuracy(GB_LIST))THEN
```

```

GB_LIST ← IB_LIST;
ENDIF
t = t + 1;
ENDWHILE
RETURN (GB_LIST);
    
```

Algorithm 3 presents the high-level pseudo code of the Pittsburgh strategy. The strategy works as follows. An ant in the colony (corresponding to an iteration of the for loop) starts with an empty list of rules and adds one rule at a time to that list while the number of uncovered training examples is greater than a user-specified maximum value. After a rule is created and pruned, the training examples covered by the rule are removed and the rule is added to the current list of rules. Note that the heuristic information is recalculated at each iteration of the list creation process WHILE loop in order to reflect the potential changes in the predictive power of the terms due to the removal of training examples covered by previous rules. When an ant finishes the list creation process, the iteration best list is updated if the quality of the newly created list is greater than the quality of the iteration best list. After all ants create a candidate list of rules, pheromone values are updated using the iteration best list of rules and the global-best list of rules is updated, if the quality of the iteration best list is greater than the quality of the global-best list.

In order to use pheromone to create multiple rules covering different set of training examples, the pheromone matrix is extended to include a *tour* identification, which corresponds to the number of the rule being created (e.g., 1 for the first rule, 2 for the second rule, and so forth). Each entry in the pheromone matrix corresponding to an edge of the construction graph is represented not just by a pair (v_i, v_j) where v_i and v_j correspond to the vertices connected by e_{ij} but rather it is represented by a triplet $(tour, v_i, v_j)$. This way, an ant will use the pheromone entries corresponding to the number of the rule (*tour*) being created during the rule construction process. The probability of an ant to follow the edge leading to a vertex v_j when creating the rule t and located at vertex v_i is given by:

$$P_{v_j} = \frac{\tau_{(t, v_i, v_j)} * \eta_{V_j}}{\sum_{k=1}^{F_{V_i}} \tau_{(t, v_i, v_k)} * \eta_{V_k}} \quad (5)$$

where $\tau_{(t, v_i, v_j)}$ is the amount of pheromone associated with the entry (t, v_i, v_j) in the pheromone matrix, η_{V_k} is the heuristic information associated with vertex v_k and F_{V_i} is the set of neighbour vertices of vertex v_i .

The pheromone update also takes into account the tour identification and the update procedure is accomplished in two steps. Firstly, pheromone evaporation is simulated by decreasing the amount of pheromone of each entry by a user-defined factor. Secondly, the amount of pheromone of the entries used in the iteration best list of rules is increased based on the quality of the list of rules, which corresponds to its predictive accuracy measured on the training set. The pheromone update rule is given by:

$$\tau_{(t, v_i, v_j)} = \begin{cases} \rho \cdot \tau_{(t, v_i, v_j)}, & \text{if } (t, v_i, v_j) \notin IB_LIST \\ \rho \cdot \tau_{(t, v_i, v_j)} + Q(IB_LIST), & \text{if } (t, v_i, v_j) \in IB_LIST \end{cases} \quad (6)$$

where ρ is the evaporation factor, $\tau_{(t,v_i,v_j)}$ is the amount of pheromone associated with the entry (t, v_i, v_j) , t is the tour identification (i.e., the number of rule where the edge between vertices v_i and v_j was used), v_i is the start vertex of the edge and v_j is the end vertex of the edge and $Q(GB_LIST)$ is the quality of the iteration-best list of rules, measured as the predictive accuracy (number of correct predictions divided by the total number of predictions) in the training set. The values given by Equation (6) are limited to the interval τ_{MIN}, τ_{MAX} , following the same approach as the MAX–MIN Ant System (MMAS). MMAS imposes explicit limits τ_{MIN} and τ_{MAX} on the minimum and maximum pheromone values to constrain all pheromone values $\tau_{(t,v_i,v_j)}$ to the range $\tau_{MIN} \leq \tau_{(t,v_i,v_j)} \leq \tau_{MAX}$. These limits are dynamically updated each time a new best solution is found. Additionally, the τ_{MIN} and τ_{MAX} values are also used to determine the stagnation of the search. When all edges followed by the ant that created the iteration best list of rules are associated with τ_{MAX} and the remaining edges are associated with τ_{MIN} , the search has become stagnant and the algorithm stops. The rule construction process, pruning procedure, and heuristic information of Ant Miner-PB are based on Ant Miner-MC. Ant Miner-PB algorithms like Ant Miner-MC algorithm can cope with both nominal and continuous attributes, unlike the original Ant-Miner algorithm, which can cope with nominal attributes only.

An important characteristic of the sequential covering strategy is that there is no pre-defined number of rules required to create a candidate list of rules and ants have the flexibility of creating lists of different lengths. The number of rules that an ant creates depends on the available training examples at each iteration of the list creation process (inner WHILE loop in Algorithm 3), which varies according to examples covered by the previous rules created by the ant. The use of a different set of pheromone values for each rule they are creating indirectly encodes the order (sequence) that ants create the rules, which represents the interaction between them. This highlights the main difference from algorithm based on Michigan approach. The aim of the algorithm is to converge to the best list of rules, instead of converging to the list of best rules.

7. OUR PROPOSED WORK BASED ON PITTSBURGH APPROACH

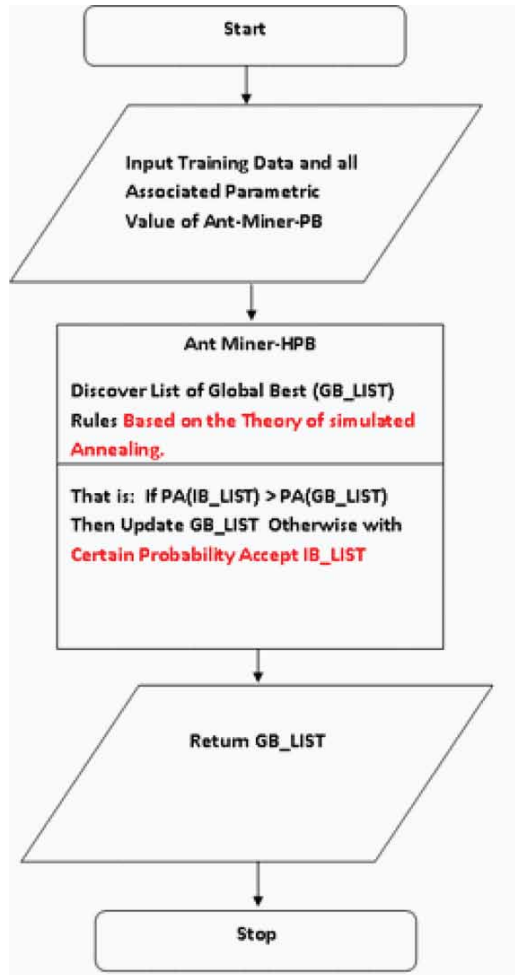
This work overcomes the problem of rule interaction (i.e., the outcome of a rule affects the rules that can be discovered subsequently since the search space is modified due to the removal of instances covered by previous rules) and reduces the local optimality by hybridizing ant miner based on Pittsburgh style approach (Ant Miner-HPB) and simulated annealing. A pictorial representation in a broader aspect of our approach is given in Figure 1. The high-level description of this approach is given in Algorithm 4.

Algorithm 4. Hybridized Ant Miner based on Pittsburgh approach (Ant Miner-HPB)

```

Procedure_AntMiner_Pittsburgh_S.A.Style_HP()
Input : TrainingSet ;
Output : ListofRules ;
ComputationalSteps :
INITIALIZATION_OF_PHEROMONES() ;
COMPUTATION_OF_HEURISTIC_INFORMATION() ;
K = 1; K_LIST = PRUNE(CRETERULE(TrainingSet));
GB_LIST ← K_LIST ;
    
```


Figure 1. Flow diagram of Ant-Miner-HPB



```

T = TMAX ;
WHILE (T ≥ TMIN AND NOT STAGNATION) DO
  IB_LIST ← ∅ ;
  FOR i ← 1 to Size(COLONY) DO
    M ← Size(TrainingSet);
    i_LIST ← ∅;
    WHILE (M > Maximum_Uncovered) DO
      COMPUTATION_OF_HEURISTIC_INFORMATION(TrainingSet) ;
      RULE ← PRUNE(CRETERULE(TrainingSet)) ;
      TrainingSet ← TrainingSet - Covered(RULE, TrainingSet);
      i_LIST ← i_LIST + RULE
    ENDWHILE
  
```

```

IF (Predictive_Accuracy(i_LIST)) > (Predictive_Accuracy(IB_RULE))
IB_LIST ← i_LIST ;
ENDIF
ENDFOR
Update_Pheromones(IB_LIST);
IF (Predictive_Accuracy(GB_LIST)) > (Predictive_Accuracy(IB_LIST))
IF (Predictive_Accuracy(IB_LIST)) > (Predictive_Accuracy(K_LIST))
K + 1_LIST ← IB_LIST
ELSE
GenerateaRandomUniformnumberUK;
IFUK < exp  $\left( - \frac{(\text{Predictive\_Accuracy}(\text{IB\_LIST})) - (\text{Predictive\_Accuracy}(\text{K\_LIST}))}{T} \right)$ 
K + 1_LIST ← IB_LIST ;
ELSE
K + 1_LIST ← K_LIST ;
ENDIF
ENDIF
ELSE
GB_LIST ← IB_LIST;
ENDIF
T = α * T;
K ← K + 1;
ENDWHILE
RETURN(GB_LIST);

```

The sub-routine of *PRUNE()* and *CREATERULE(TrainingSet)* are described as follows:

```

PRUNE(LIST)
LIST ← CREATERULE(TrainingSet);
PRUNE(LIST) ;
RETURN

CREATERULE(TrainingSet)
list ← ∅ ;
WHILE (M > Maximum_Uncovered) DO
    RULE ← PRUNE(CREATERULE(TrainingSet));
    TrainingSet ← TrainingSet - Covered(RULE, TrainingSet);
    LIST = LIST + RULE;
ENDWHILE
RETURNLIST ;

```

8. EXPERIMENTAL WORK

In this Section, we provide the description of the datasets, environment, parameter setup, and finally presents the results obtained from our proposed method.

9. DATASET DESCRIPTION

To validate the proposed method, we have used six public domain datasets retrieved from the University of California at Irvine (UCI) machine learning repository. The datasets involve binary (two class values) and multiclass (more than two class values) classification problems, with both nominal and continuous predictor attributes. Table 1 presents a summary of the datasets used in the experiment. The first column of this table gives the name of datasets, while the other columns indicate, respectively, the number of cases, the number of categorical attributes, the number of continuous attributes, and the number of classes of the dataset.

The motivation of considering the aforesaid dataset is to compare our proposed method with the works presented in (Parpinelli, Lopes & Freitas, 2002).

10. PERFORMANCE METRIC AND PARAMETER SET UP

We use predictive accuracy as a main performance metric. It is defined as the percentage of testing samples correctly classified by the classifier. This is a popular performance metric for general comparison between classification algorithms and has been used in all ant miner approaches. The experiments are performed using a standard 10 fold cross validation procedure. In this procedure, the test cases are divided into 10 fold (i.e., 10 equally sized mutually exclusive subsets). Each of the subset is used once for testing while the other nine are used for training. The results of the 10 runs are then averaged and this average is reported as the final result along with standard deviation. The parameters of our approach Ant Miner-HPB like Ant Miner-PB are distinctly setting up as follows.

11. PARAMETERS OF ANT MINER-HPB

The following fixed parameter values were used for all experimental runs in the Pittsburgh approach:

Max_iteration ← Maximum Temperature $T_{max} = 1500$

Table 1. Description of datasets

Dataset	#Cases	#Categorical Attribute	#Continuous Attribute	Classes
Ljubljana breast cancer	286	9	-	2
Wisconsin breast cancer	699	-	9	2
Tic-tac-toe	958	9	-	2
Dermatology	366	33	1	6
Hepatitis	155	13	6	2
Cleve and heart disease	303	8	5	5

Size_Colony ← The total number of ants= 5

Evaporation_factor ← Maximum and Minimum pheromone value= 90

Min_cases_per_rule ← Minimum number of cases per rule = 10

Max_uncovered_cases ← Maximum number of uncovered cases in the training set=10

Since the values of minimum number of cases and maximum uncovered parameters are related, i.e., the maximum uncovered should be at least the same as the minimum number of cases. We have used the same value for both, specified by the minimum number of cases parameter. The value of α is set as 0.998.

12. RESULTS AND ANALYSIS

We have evaluated the performance of Ant Miner-HPB and compare it with Ant Miner-PB. All the results of the comparison were obtained using an Intel PC with 1 GB of main memory. Ant Miner-HPB was developed in MATLAB6.5 version and it took about a little more processing time as compared to Ant Miner-PB (on the order of seconds for each dataset) to obtain the results.

The comparison was carried out across two criteria, namely the predictive accuracy of the discovered rule lists and their simplicity. Predictive accuracy was measured by a well-known 10-fold cross-validation procedure. The predictive accuracies (on the test set) of the 10 runs are then averaged and reported as the predictive accuracy of the discovered rule list.

The results comparing the predictive accuracy of Ant Miner-HPB and Ant Miner-PB are reported in Table 2. The numbers right after the “±” symbol is the standard deviations of the corresponding predictive accuracies rates. As shown in this Table 2, Ant Miner-HPB discovered rules with a better predictive accuracy than Ant Miner-PB in all data sets, Ant Miner-HPB was significantly more accurate than the Ant Miner-PB, that is, the corresponding predictive accuracy intervals (taking into account the standard deviations) do not overlap.

Similarly results concerning the simplicity of the discovered rule list, measured, as usual in the literature, by the number of discovered rules and the average number of terms (conditions) per rule. The results comparing the simplicity of the rule lists discovered by Ant Miner-HPB and Ant Miner-PB are reported in Table 3. An important observation is that for Ljubljana breast cancer, Wisconsin breast cancer and Hepatitis datasets, the rule list discovered by Ant Miner-PB was simpler then rule list discovered by Ant Miner-HPB. In the tic-tac-toe, Dermatology and Cleve and heart disease data set, Ant Miner-HPB discovered a rule list on the order of 2 times lesser than Ant Miner PB.

Table 2. Average predictive accuracy of Ant Miner-HPB in %, measured by 10-fold cross-validation

Data Set	Ant Miner-PB	Ant Miner-HPB
Ljubljana breast cancer	72.32 ± 0.31	78.16±0.87
Wisconsin breast cancer	94.29 ± 0.16	97.71±1.87
Tic-tac-toe	74.89±0.27	98.22±1.83
Dermatology	92.46 ± 0.31	95.21±3.45
Hepatitis	66.72 ± 0.40	95.78±5.26
Cleve and heart disease	55.50 ± 0.37	59.38±2.16

Table 3. Average number of terms (simplicity) in the discovered list of Ant Miner-HPB measured by 10-fold cross-validation

Data Set	Ant Miner-PB	Ant Miner-HPB
Ljubljana breast cancer	19.15 ± 0.40	21.28±0.38
Wisconsin breast cancer	8.55 ± 0.12	11.38±0.68
Tic-tac-toe	17.23±0.08	7.68±0.79
Dermatology	44.47 ± 0.63	21.27±0.03
Hepatitis	11.78 ± 0.08	11.88±0.08
Cleve and heart disease	27.65 ± 0.58	14.37±0.01

Taking into account both the predictive accuracy and rule list simplicity criteria, the results of our experiments can be summarized as follows. Concerning classification accuracy, Ant Miner-HPB obtained results are significantly better than Ant Miner-PB in all of the six data sets. Concerning the simplicity of discovered rules, overall Ant Miner-HPB discovered rule lists relatively simpler (smaller) than the rule lists discovered by Ant Miner-PB. This seems a good trade-off, since in many data mining applications the simplicity of a rule list/set tends to be even more important than its predictive accuracy.

13. CONCLUSION

In this paper, we present a novel variant of ACO based classification algorithm, which is based on Pittsburgh approach. The novel algorithm is inspired and conceived by considering the best attributes of ACO based classification rule mining and simulated annealing (SA). Incorporating SA in ACO based classification is straight forward. Four ingredients are needed: i) a concise description of a configuration of the system; ii) a random generator of moves on rearrangements of the elements in a configuration; iii) a quantitative objective function containing the trade-offs that have to be made; and iv) an annealing schedule of the temperature. The annealing schedule may be developed by trial and error for a given problem, or may consist of just warming the system until it is obviously melted, then cooling in slow stages until diffusion of the components ceases. Inventing the most effective sets of moves and deciding which factors to incorporate into the objective function require insight into problem being solved and may not be obvious. However, existing methods of iterative improvement can provide natural elements in which to base a simulated annealing algorithm. The experimental results indicated that our algorithms have potential to compete with other state-of-art contemporary classification algorithms. The main advantage of our approach was high accuracy combined with comprehensibility of the discovered rule sets.

REFERENCES

- Aarts, E., & Korst, J. (1989). *Simulated Annealing and Boltzmann Machines*. John Wiley and Sons.
- Abraham, A., Grosan, C., & Ramos, V. (2006). Swarm Intelligence in Data Mining. In A. Abraham, C. Grosan & V. Ramos (Eds.), *Swarm Intelligence in Data Mining*. Studies in Computational Intelligence. Springer. doi:10.1007/978-3-540-34956-3
- Al-Behadili, H. N. K. (2018). Intelligent Hypothermia Care System using Ant Colony Optimization for Rules Prediction. *Journal of University of Babylon*, 26(2), 47–46.
- Ali, Z., & Shahzad, W. (2017). Comparative analysis and survey of ant colony optimization based rule miner. *International Journal of Advanced Computer Science and Applications*, 8(1), 48–60. doi:10.14569/IJACSA.2017.080108
- Angus, D., & Woodward, C. (2009). Multiple objective ant colony optimisations. *Swarm Intelligence*, 3(1), 69–85. doi:10.1007/s11721-008-0022-4
- Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4), 353–373. doi:10.1016/j.plprev.2005.10.001
- Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.
- Dehuri, S., Ghosh, A., & Ghosh, S. (2008). *Multi-objective Evolutionary Algorithm for Knowledge Discovery from Database*. Springer.
- Dehuri, S., & Mall, R. (2006). Predictive and Comprehensive rule discovery using a multi-objective genetic algorithm. *Knowledge-Based Systems*, 19(6), 413–421. doi:10.1016/j.knosys.2006.03.004
- Dehuri, S., Patnaik, S., Ghosh, A., & Mall, R. (2008). Application of Ellitist Multi-Objective Genetic Algorithm for Classification Rule Generation. *Applied Soft Computing*, 8(1), 477–487. doi:10.1016/j.asoc.2007.02.009
- Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3), 243–278. doi:10.1016/j.tcs.2005.05.020
- Dorigo, M., Maniezzo, V., & Colnari, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems, Man and Cybernetics Part B*, 26(1), 29–41.
- Dorigo, M., & Stutzle, T. (2006). *Ant Colony Optimization*. New Delhi: Prentice-Hall of India Private Ltd.
- Freitas, A. A. (2002). *Data mining and knowledge discovery with evolutionary algorithms*. Springer-Verlag. doi:10.1007/978-3-662-04923-5
- Han, J., & Kamber, M. (2006). *Data Mining: Concepts & Techniques* (2nd ed.). Morgan Kaufmann Publishers.
- Hwang, C.-R. (1988). Simulated annealing: Theory and applications. *Acta Applicandae Mathematicae*, 12(1), 108–111.
- Kalia, H., Dehuri, S., Ghosh, A., & Cho, S.-B. (2018). Surrogate-assisted multi-objective genetic algorithm for fuzzy rule based classification. *International Journal of Fuzzy Systems*, 20(6), 1938–1955. doi:10.1007/s40815-018-0478-3
- Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated Annealing Theory and Application*. Dordrecht: Reidel. doi:10.1007/978-94-015-7744-1
- Lim, C. P., Jain, L. C., & Dehuri, S. (2009). *Innovations in Swarm Intelligence*. Studies in Computational Intelligence, 248. Springer. doi:10.1007/978-3-642-04225-6
- Liu, B., Abbass, H. A., & McKay, B. (2004). Classification rule discovery with ant colony optimization. *IEEE Computational Intelligence Bulletin*, 3(1), 65–76.
- Liu, Y., Zhang, Q., & Yu, L. (2019). Picking robot path planning based on improved ant colony algorithm. *Proceedings of the 2019 34th Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 473–478). IEEE. doi:10.1109/YAC.2019.8787592

- Mahapatra, B., & Patnaik, S. (2018). Ant Colony Optimization. In *Advances in Swarm Intelligence for Optimizing Problems in Computer Science* (pp. 79-114). Chapman and Hall/CRC. doi:10.1201/9780429445927-4
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. H., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1087–1092. doi:10.1063/1.1699114
- Otero, F., Freitas, A., & Johnson, C. (2008). Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. *Proceedings of the 6th International Conference on Swarm Intelligence (ANTS 2008)* (pp. 48–59). Springer Verlag. doi:10.1007/978-3-540-87527-7_5
- Panda, M., Dehuri, S., & Patra, M. R. (2015). *Modern approach of data mining: Theory and practice*. Alpha Science International Ltd.
- Parpinelli, R., Lopes, H., & Freitas, A. (2001). An ant colony based system for data mining: applications to medical data. *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 791–798). Academic Press.
- Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4), 321–332. doi:10.1109/TEVC.2002.802452
- Tao, Y. (2018). Swarm intelligence in humans: A perspective of emergent evolution. *Physica A*, 502, 436–446. doi:10.1016/j.physa.2018.02.120

Bijaya Kumar Nanda is working as an Assistant professor in the Department of Mathematics, BIET, Bhadrak, Odisha since 2001. He obtained his M. Tech (Computer Science) from Utkal university and Ph.D. in Computer Science from Fakir Mohan University Odisha in the year 2003 and 2016, respectively. His area of interest includes data mining, soft computing, and multi objective optimization.

Satchidananda Dehuri is a Professor of the faculty of Department of Information and Communication Technology, and Chairman, Post Graduate Council of Fakir Mohan University, Vyasa Vihar, Balasore. He received M. Tech. and Ph.D. degrees in Computer Science from Utkal University, Vani Vihar, Odisha in 2001 and 2006, respectively. He completed Post-Doctoral Research in Soft Computing Laboratory, Yonsei University, Seoul, South Korea under the BOYSCAST Fellowship Program of DST, Govt. of India. In 2010, he received the Young Scientist Award in Engineering and Technology for the year 2008 from Odisha Vigyan Academy, Department of Science and Technology, Govt. of Odisha. He has published about 200 research papers in reputed journals and referred conferences, has published four textbooks for undergraduate and post graduate students and edited 10 books. As a part of the Academic Collaboration, he has visited Ireland, New Zealand, Hong Kong, and France.