A new move towards updating pheromone trail in order to gain increased predictive accuracy in classification rule mining by implementing ACO algorithm

Vimal G.Bhatt[1] Priyanka Trikha[2]
[1] M.Tech Student, Computer Science Department
[2]Assistant Professor, Computer Science Department
[1,2] Sri Balaji College of Engineering &Technology, Jaipur, Rajasthan

Abstract— Ant miner algorithm is used to find the classification rule which helps to do classification of the data. Ant miner uses the Ant Colony Optimization (ACO) which deals with artificial systems that is inspired from the foraging behavior of real ants. Here the task is to improve the pheromone update method in the current system. Pheromone updating is dependent mainly on the initial pheromone of the term, and term Q (quality of term) which is added to current accumulated pheromone. In this methods a try is made to lay pheromone on trail such that selection of terms is not biased and unified behavior for the system as a whole, produce a robust system capable of finding high-quality solutions for problems. Here in this approach amount of pheromone added with the used term is not directly dependent on accumulated pheromone but also on the Q (quality of rule). So here in Q is modified and multiplied in ways that help to get better solution. For this use of rule length is done and manipulated in ways that support approach to achieve goal. Thus, aim is to improve the accuracy and sustaining rule list simplicity using Ant Colony Optimization in data mining.

Keywords: - Aco Algorithm, Current Algorithm, (C-Ant miner) Proposed Algorithm (New Pheromone Update Algorithm)

I. INTRODUCTION

Mining information and knowledge from large database has been recognized by many researchers as a key research topic in database system and machine learning and by many industrial companies as an important area with an opportunity of major revenues. One of the data mining tasks gaining significant attention is the classification rules extraction from databases. The goal of this task is to assign each case (object, record or instance) to one class, out of a set of predefined classes, based on the values of some attributes for the case. There are different classifications algorithms use to extract relevant relationship in the data as decision trees which operate performing a successive partitioning of cases until all subsets belong to single class. There have been many approaches for data classification, such as statistical and roughest approaches and neural networks. Though these classification techniques are algorithmically strong they require significant expertise to work effectively and do not provide intelligible rules. The classification problem becomes very hard when the number of possible different combinations of parameters is so high that algorithms based on exhaustive searches of the parameter space rapidly become computationally infeasible. In recent years, Ant Colony System (ACS) algorithm (Dorigo & Maniezzo, 1996) has emerged as a promising technique to discover useful and interesting knowledge from database.

II. ANT COLONY OPTIMIZATION

In a series of experiments on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route. A model explaining this behavior is as follows:

1. An ant (called "blitz") runs more or less at random around the colony;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail of pheromone;
3. These pheromones are attractive, nearby ants will be inclined to follow, more or less directly, the track;
4. Returning to the colony, these ants will strengthen the route;
5. If there are two routes to reach the same food source then, in a given amount of time, the shorter one will be travelled by more ants than the long route;
6. The short route will be increasingly enhanced, and therefore become more attractive;
7. The long route will eventually disappear because pheromones are volatile;
8. Eventually, all the ants have determined and therefore "chosen" the shortest route.

The idea of ACO is to let artificial ants construct solutions for a given combinatorial optimization problem. A prerequisite for designing an ACO algorithm is to have a constructive method which can be used by an ant to create different solutions through a sequence of decisions. Typically an ant constructs a solution by a sequence of probabilistic decisions where every decision extends a partial solution by adding a new solution component until a complete paper.

ACO scheme:
Initialize pheromone values
repeat
for ant k ∈ {1, . . .,m} construct a solution
endfor
forall pheromone values do
decrease the value by a certain percentage {evaporation}

All rights reserved by www.ijsrd.com

endfor forall pheromone values corresponding to good solutions do increase the value {intensification} endfor until stopping criterion is met

III. CURRENT SYSTEM

3.1. Algorithm of current system

In this section we discuss in detail Ant Colony Optimization algorithm for the discovery of classification rules, called C-Ant-Miner [3].

Input parameter:

1. Number of ants (No_of_ants)
2. Minimum number of cases per rule (Min_cases_per_rule)
3. Maximum number of uncovered cases in the training set (Max_uncovered_cases)
4. Number of rules used to test convergence of the ants (No_rules_converg)

Existing C-Ant miner algorithm

procedure C-ant miner
begin
While stopping criterion not satisfied do
Initialize the pheromone
Repeat
Start ant with empty rule and starts rule construction.
for each
ant do
Choose next term to be added in a rule
End
Remove irrelevant terms that are added during the rule construction.
Update the pheromone(accumulated pheromone is multiplied by Q)
end repeat
Choose best rule from constructed rules
Add best rule to the discovered rule list
Reinitialize training set
End
while
end

OUTPUT: Classification rule that will classify data according to the predetermined class

A General Description of C-Ant Miner In an ACO algorithm each ant incrementally constructs/modifies a solution for the target problem. Here the target problem is the discovery of classification rules. As discussed in the introduction, each classification rule has the form: IF < term1 AND term2 AND ...> THEN <class> Each term is a triple <attribute, operator, value>, where value is a value belonging to the domain of attribute. The operator element in the triple is a relational operator. This version of Ant-Miner copes only with categorical attributes, so that the operator element in the triple is always "=". Continuous (real-valued) attributes are discretized in a preprocessing step.

A Pseudo code for Existing C-Ant miner algorithm is mention above. Ant-Miner follows a sequential covering approach to discover a list of classification rules covering all, or almost all, the training cases. At first, the list of discovered rules is empty and the training set consists of all the training cases. Each iteration of the WHILE loop of pseudo code, corresponding to a number of executions of the REPEAT-UNTIL loop, discovers one classification rule. This rule is added to the list of discovered rules, and the training cases that are correctly covered by this rule (i.e., cases satisfying the rule antecedent and having the class predicted by the rule consequent) are removed from the training set. This process is iteratively performed while the number of uncovered training cases is greater than a user-specified threshold, called Max_uncovered_cases. Each iteration of the REPEAT-UNTIL loop of pseudo code consists of three steps, comprising rule construction, rule pruning, and pheromone updating, detailed as follows. First, Antt starts with an empty rule, that is, a rule with no term in its antecedent, and adds one term at a time to its current partial rule. The current partial rule constructed by an ant corresponds to the current partial path followed by that ant. Similarly, the choice of a term to be added to the current partial rule corresponds to the choice of the direction in which the current path will be extended. The choice of the term to be added to the current partial rule depends on both a problem-dependent heuristic function (η) and on the amount of pheromone (τ) associated with each term, as will be discussed in detail in the next subsections. Antt keeps adding one-term-at a time to its current partial rule until one of the following two stopping criteria is met:

1. Any term to be added to the rule would make the rule cover a number of cases smaller than a user-specified threshold, called Min_cases_per_rule (minimum number of cases covered per rule).
2. All attributes have already been used by the ant, so that there is no more attributes to be added to the rule antecedent. Note that each attribute can occur only once in each rule, to avoid invalid rules such as "IF (Sex= male) AND (Sex = female) ..."

Second, rule Rt constructed by Antt is pruned in order to remove irrelevant terms, as will be discussed later. For the moment, we only mention that these irrelevant terms may have been included in the rule due to stochastic variations in the term selection procedure and/or due to the use of a shortsighted, local heuristic function – which considers only one-attribute-at-a-time, ignoring attribute interactions. Third, the amount of pheromone in each trail is updated, increasing the pheromone in the trail followed by Antt (according to the quality of rule Rt) and decreasing the pheromone in the other trails (simulating the pheromone evaporation). Then another ant starts to construct its rule, using the new amounts of pheromone to guide its search. This process is repeated until one of the following two conditions is met:

1. The number of constructed rules is equal to or greater than the user-specified threshold No_of_ants.
2. The current Antt has constructed a rule that is exactly the same as the rule constructed by the previous No_rules_converg–1 ants, where No_rules_converg stands for the number of rules used to test convergence of the ants. Once the REPEAT-UNTIL loop is completed, the best rule

All rights reserved by www.ijsrd.com

among the rules constructed by all ants is added to the list of discovered rules, as mentioned earlier, and the system starts a new iteration of the WHILE loop, by reinitializing all trails with the same amount of pheromone.

Detail description of each phase of algorithm.

A) Pheromone Initialization

A termij corresponds to a segment in some path that can be followed by an ant. When the first ant starts its search, all paths have the same amount of pheromone. The initial amount of pheromone deposited at each path is inversely proportional to the number of values of all attributes, and is defined by

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i} \qquad (1)$$

'a' is the total number of attributes, 'bi' is the number of values in the domain of attribute i.

The value returned by this equation is normalized to facilitate its use in 8.

B) Discretization of continuous value Dynamic Discretization of continuous value is done by Entropy based discretization method.

If nominal attribute, where every termij has the form (ai = vij), the entropy for the attribute-value pair is computed as in (2) – used in the original

$$ep_v(a_i) \equiv \frac{|S_{ai<v}|}{|S|}.entropy(a_i < v) + \frac{|S_{ai>v}|}{|S|}.entropy(a_i \geq v) \qquad (2)$$

Ant-Miner: (2) where, p(c | ai = vij) is the empirical probability of observing class c conditional on having observed ai= vij k is the number of classes. To compute the entropy of nodes representing continuous attributes (termi) since these nodes do not represent an attribute-value pair, we need to select a threshold value v to dynamically partition the continuous attribute ai into two intervals: a < v and ai >= v. The best threshold value is the value v that minimizes the entropy of the partition, given by (8):

$$ep_v(a_i) \equiv \frac{|S_{ai<v}|}{|S|}.entropy(a_i < v) + \frac{|S_{ai>v}|}{|S|}.entropy(a_i \geq v) \quad (3)$$

where, |Sai<v| is the total number of examples in the partition ai < v (partition of training examples where the attribute ai has a value less than v) |Sai>=v| is the total number of examples in the partition ai>= v (partition of training examples where the attribute ai has a value greater or equal to v) |S| is the total number of training examples.

After the selection of the threshold vbest, the entropy of the termi corresponds to the minimum entropyvalue ofthe two partitions and it is definedas:[3] en (t

$$entropy(term_i) \equiv \min(entropy(a_i < v_{best}), entropy(a_i \geq v_{best})) \qquad (4)$$

Term added to current partial rule Let a termij be a rule condition of the form Ai = Vij, where Ai is the i-th attribute and Vij is the j-th value of the domain of Ai. The probability that termij is chosen to be added to the current partial rule is given by Equation: $PPiiii$=⍰(τij(t

$$P_{ij} = \frac{(\eta_{ij})(\tau_{ij}(t))}{\sum_{i=1}^{a} xi \sum_{i=1}^{bi}((\eta_{ii})(\tau_{ii}(t)))} \qquad (5)$$

where: ηij is the value of a problem-dependent heuristic function for termij. The higher the value of ηij, the more relevant for classification the termij is, and so the higher its probability of being chosen. Problem-dependent heuristic function will be discussed in the next sub section(c) of this section. τij(t) is the amount of pheromone associated with termij at iteration t, corresponding to the amount of pheromone currently available in the position i,j of the path being followed by the current ant. The better the quality of the rule constructed by an ant, the higher the amount of pheromone added to the trail segments visited by the ant. Therefore, as time goes by, the best trail segments to be followed – that is, the best terms (attribute-value pairs) to be added to a rule – will have greater and greater amounts of pheromone, increasing their probability of being chosen. a is the total number of attributes. xi is set to 1 if the attribute Ai was not yet used by the current ant, or to 0 otherwise. bi is the number of values in the domain of the i-th attribute. D) Heuristic Function

For each termij that can be added to the current rule, Ant-Miner computes the value ηij of a heuristic function that is an estimate of the quality of this term, with respect to

its ability to improve the predictive accuracy of the rule. More precisely, the value of ηij for termij involves a measure of the entropy (or amount of information) associated with that term. For each termij of the form Ai=Vij – where Ai is the i-th attribute and Vij is the j-th value belonging to the domain of

$$H\left(W/_{A_i = V_{ij}}\right) = -\sum_{w=1}^{k}\left(P\left(W/_{A_i = V_{ij}}\right).log_2 k\left(P\left(W/_{A_i = V_{ij}}\right)\right)\right) \qquad (6)$$

where: W is the class attribute (i.e., the attribute whose domain consists of the classes to be predicted). k is the number of classes. P(w| Ai =Vij) is the empirical probability of observing class w conditional on having observed Ai =Vij. The higher the value of H(W| Ai =Vij), the more uniformly distributed the classes are and so, the smaller the probability that the current ant chooses to add termij to its partial rule. It is desirable to normalize the value of the heuristic function to facilitate its use in 8. In order to implement this normalization, it is used the fact that the value of H(W| Ai =Vij) varies in the range 0 ≤ H(W| Ai =Vij) ≤ log2 k, where k is the number of classes. Therefore, the proposed normalized, information-theoretic heuristic function is:

$$\eta_{ij} = \frac{(\log_2 k) - H\left(W/_{Ai = Vij}\right)}{\sum_{i=1}^{a} xi \sum_{j=1}^{bi}\left((\log_2 k) - H\left(W/_{Ai = Vij}\right)\right)} \qquad (7)$$

where a, xi, and bi have the same meaning as in 8. Note that the H(W| Ai =Vij) of termij is always the same, regardless of the contents of the rule in which the term occurs.

All rights reserved by www.ijsrd.com

Therefore, in order to save computational time, the H(W| Ai =Vij) of all termij is computed as a preprocessing step.

In the above heuristic function there are just two minor caveats. First, if the value Vij of attribute Ai does not occur in the training set then H(W| Ai =Vij) is set to its maximum value of log2k. This corresponds to assigning to termij the lowest possible predictive power. Second, if all the cases belong to the same class then H(W| Ai =Vij) is set to 0. This corresponds to assigning to termij the highest possible predictive power. In Ant-Miner the entropy is computed for an attribute value pair only, since an attribute-value pair is chosen to expand the rule. The entropy measure is heuristic function together with pheromone updating. This makes the rule-construction process of Ant-Miner more robust and less prone to get trapped into local optima in the search space, since the feedback provided by pheromone updating helps to correct some mistakes made by the shortsightedness of the entropy measure. Note that the entropy measure is a local heuristic measure, which considers only one attribute at a time, and so is sensitive to attribute interaction problems. In contrast, pheromone updating tends to cope better with attribute interactions, since pheromone updating is directly based on the performance of the rule as a whole (which directly takes into account interactions among all attributes occurring in the rule). E) Rule Pruning The main goal of rule pruning is to remove irrelevant terms that might have been unduly included in the rule. Rule pruning potentially increases the predictive power of the rule, helping to avoid its over fitting to the training data. Another motivation for rule pruning is that it improves the simplicity of the rule, since a shorter rule is usually easier to be understood by the user than a longer one. As soon as the current ant completes the construction of its rule, the rule pruning procedure is called. The basic idea of rule pruning is to iteratively remove one-term-at-a-time from the rule while this process improves the quality of the rule. More precisely, in the first iteration one starts with the full rule. Then it is tentatively tried to remove each of the terms of the rule – each one in turn – and the quality of the resulting rule is computed using a given rule-quality function as:

$$Q = \left(\frac{TP}{TP + FN}\right) \cdot \left(\frac{TN}{FP + TN}\right) \qquad (8)$$

TP (true positives) is the number of cases covered by the rule that have the class predicted by the rule.

FP (false positives) is the number of cases covered by the rule that have a class different from the class predicted by the rule. FN (false negatives) is the number of cases that are not covered by the rule but that have the class predicted by the rule. TN (true negatives) is the number of cases that are not covered by the rule and that do not have the class predicted by the rule. Q´s value is within the range $0 \leq Q \leq 1$ and, the larger the value of Q, the higher the quality of the rule. It should be noted that this step might involve replacing the class in the rule consequent, since the majority class in the cases covered by the pruned rule can be different from the majority class in the cases covered by the original rule. The term whose removal most improves the quality of the rule is effectively removed from it, completing the first iteration. In the next iteration it is removed again the term whose removal most improves the quality of the rule, and so on. This process is repeated until the rule has just one term or until there is no term whose removal will improve the quality of the rule. E) Pheromone Updating The initial amount of pheromone deposited at each path position is inversely proportional to the number of values of all attributes, and is defined by (1). Whenever an ant constructs its rule and that rule is pruned (see pseudo code) the amount of pheromone in all segments of all paths must be updated. This pheromone updating is supported- by two basic ideas, namely:

1. The amount of pheromone associated with each termij occurring in the rule found by the ant (after pruning) is increased in proportion to the quality of that rule.
2. The amount of pheromone associated with each termij that does not occur in the rule is decreased, simulating pheromone evaporation in real ant colonies.

1) Increasing the Pheromone of Used Terms

Increasing the amount of pheromone associated with each termij occurring in the rule found by an ant corresponds to increasing the amount of pheromone along the path completed by the ant. In a rule discovery context, this corresponds to increasing the robability of termij being chosen by other ants in the future in proportion to the quality of the rule Pheromone updating for a termij is performed according to Equation (9), for all terms termij that

$$\tau ij(t+1) = \tau ij(t) + \tau ij(t).Q \quad \forall i,j \in R \qquad (9)$$

occur in the rule. Where R is the set of terms occurring in the rule constructed by the ant at iteration t. Therefore, for all termij occurring in the rule found by the current ant, the amount of pheromone is increased by a fraction of the current amount of pheromone, and this fraction is given by Q(according to (8)).

2) Decreasing the Pheromone of Unused Terms As mentioned above, the amount of pheromone associated with each termij that does not occur in the rule found by the current ant has to be decreased in order to simulate pheromone evaporation in real ant colonies. In Ant-Miner, pheromone evaporation is implemented in a somewhat indirect way. More precisely, the effect of pheromone evaporation for unused terms is achieved by normalizing the value of each pheromone τij. This normalization is performed by dividing the value of each τij by the summation of all τij, ∀i,j. To see how this implements pheromone evaporation, remember that only the terms used by a rule have their amount of pheromone increased by Equation 9. Therefore, at normalization time the amount of pheromone of an unused term will be computed by dividing its current value (not modified by (9)) by the total summation of pheromone for all terms (which was increased as a result of applying (9) to all used terms). The final effect will be the reduction of the normalized amount of pheromone for each unused term.

Used terms will, of course, have their normalized amount of pheromone increased due to the application of (9).

All rights reserved by www.ijsrd.com

IV. PROPOSED SYSTEM

A. Introduction to system

Intelligence (SI) is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause Swarm coherent functional global patterns to emerge. SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model. Ant Colony Optimization (ACO) deals with artificial systems that is inspired from the foraging behaviour of real ants. Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to find patterns and then to validate the findings by applying the detected patterns to new subsets of data. In order to achieve this, data mining uses computational techniques from statistics, machine learning and pattern recognition. Aim was to improve the predictive accuracy and sustain/improving rule list simplicity for the existing method available with Ant Colony Optimization in data mining. Task is achieved by improving and updating the pheromone update method in the current system. Selection of terms is not biased, leading to exploration of the attributes. The classification rule construction resulting with this method would be novel. Thus, method of pheromone update is selected or done which leads to the exploration of the terms leading to better classification rule construction.

B. Proposed algorithm input parameter:

1. *Number of ants (No_of_ants):* This is also the maximum number of complete candidate rules constructed and pruned during an iteration of the WHILE loop of Algorithm , since each ant is associated with a single rule. In each iteration the best candidate rule found is considered a discovered rule. The larger *No_of_ants*, the more candidate rules are evaluated per iteration, but the slower the system is.

2. *Minimum number of cases per rule (Min_cases_per_rule):* Each rule must cover atleast *Min_cases_per_rule* cases, to enforce at least a certain degree of generality in the discovered rules. This helps to avoid an overfitting of the training data.

3. *Maximum number of uncovered cases in the training set (Max_uncovered_cases):* The process of rule discovery is iteratively performed until the number of training cases that are not covered by any discovered rule is smaller than this threshold.

4. *Number of rules used to test convergence of the ants (No_rules_converg):* If the current ant has constructed a rule that is exactly the same as the rule constructed by the previous *No_rules_converg* – 1 ants, then the system concludes that the ants have converged to a single rule (path). The current iteration of the WHILE loop of Algorithm is therefore stopped, and another iteration is started.

ALGORITHM:

TrainingSet = {all training cases};
DiscoveredRuleList = [ ]; /* rule list is initialized with an empty list */
WHILE (TrainingSet > Max_uncovered_cases)
t = 1; /* ant index */ j = 1; /* convergence test index */
Initialize all trails with the same amount of pheromone
REPEAT

Antt starts with an empty rule and incrementally constructs a classification rule Rt adding one term at a time to the current rule;

Prune rule Rt; *Update the pheromone of all trails by increasing pheromone in the trail followed by Antt (proportional to the quality of Rt)*

And decreasing pheromone in the other trails (simulating pheromone evaporation);

IF (Rt is equal to Rt – 1) /* update convergence test */
THEN j = j + 1;
ELSE j = 1;
END IF t = t + 1;
UNTIL (i≥No_of_ants) OR (j≥No_rules_converg)

Choose the best rule Rbest among all rules Rt constructed by all the ants;

Add rule Rbest to DiscoveredRuleList;

TrainingSet= TrainingSet - {set of cases correctly covered by Rbest};
END WHILE

OUTPUT:

Classification rule that will classify data according to the predetermined class

C. Description of proposed system

1) *Pheromone Initialization* A term ij corresponds to a segment in some path that can be followed by an ant. When the first ant starts its search, all paths have the same amount of pheromone.

The initial amount of pheromone deposited at each path is inversely proportional to the number of values of all attributes, and is defined by

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i} \quad (10)$$

Where, 'a' is the total number of attributes,
'bi' is the number of values in the domain of attribute ai.

2) Pheromone Updating The initial amount of pheromone deposited at each path position is inversely proportional to the number of values of all attributes, and is defined by(1)

Whenever an ant constructs its rule and that rule is pruned (see Algorithm) the amount of pheromone in all segments of all paths must be updated. This pheromone updating is supported- by two basic ideas, namely:

1. The amount of pheromone associated with each *termij* occurring in the rule found by the ant (after pruning) is increased in proportion to the quality of that rule.

2. The amount of pheromone associated with each *termij* that does not occur in the rule is decreased, simulating pheromone evaporation in real ant colonies.

3) Increasing the Pheromone of Used Terms Increasing the amount of pheromone associated with each termij occurring in the rule found by an ant corresponds to increasing the amount of pheromone along the path completed by the ant. In a rule discovery context, this corresponds to increasing

All rights reserved by www.ijsrd.com

the probability of termij being chosen by other ants in the future in proportion to the quality of the rule

– New Pheromone Updater :

Pheromone updating for a *termij* is performed according to Equation (2), for all terms *termij* that occur in the rule.

$$\tau_{ij}(t) = (1-E)*\tau_{ij}(t-1) + ((1-(1/(1+Q)))*(1-(1/n))*\tau_{ij}(t-1) \quad (11)$$

where τ ij pheromone related to the term

Q = quality of term
E = evaporation rate
n = rule length

Therefore, for all *termij* occurring in the rule found by the current ant, the amount of pheromone is increased by a fraction of the current amount of pheromone, and this fraction is given by Q.

Where,

$$Q = \left(\frac{TP}{TP + FN}\right) \cdot \left(\frac{TN}{FP + TN}\right) \quad (12)$$

Q´s value is within the range $0 \leq Q \leq 1$ and, the larger the value of *Q*, the higher the quality of the rule.

– *TP* (true positives) is the number of cases covered by the rule that have the class predicted by the rule

– *FP* (false positives) is the number of cases covered by the rule that have a class different from the class predicted by the rule.

– *FN* (false negatives) is the number of cases that are not covered by the rule but that have the class predicted by the rule.

– *TN* (true negatives) is the number of cases that are not covered by the rule and that do not have the class predicted by the rule.

V. EXPERIMENTAL ANALYSIS

The performance of the classification method is affected by several factors (like the number of attributes, input parameter, data type of attributes etc.). This chapter we will take a closer look at how different factors actually affect the performance of algorithm.

Here a comparison of results of existing C ant-miner algorithm is shown in tabular form. Proposed New pheromone updating Here the parameter for comparison is Accuracy, No. of Rules Discovered and No. of terms/ No. of rules Discovered. Data set detail used in the experiments is described in chapter 5 section 5.2. Each algorithm is run for 10 times and then average of ten run is done to measure efficiency of algorithms. The parameter setting is shown in the following table:

PARAMETER NAME	VALUE
Minimum covered examples per rule value	5
Convergence test size	10
Maximum number of iterations	1500
Maximum number of uncovered cases	10
Number of ants in the colony	60
Cross Validation Fold	10

Table 1: parameter details

For easy in the representation of experimental results, in tables and chart by New Pheromone Updater

Data Set	Existing C ant miner	New Pheromone Updator
Heart	76.5184	77.037
Hepatitis	75.2751	77.439
Ionosphere	87.0643	88.607
WDBC	94.4448	90.862

Table 2: accuracy detail

Data Set	Existing C ant miner	New Pheromone Updator
Heart	1.9369	1.768
Hepatitis	1.9728	1.806
Ionosphere	1.5036	1.379
WDBC	1.7015	1.125

Table 3: terms per rule detail

Data Set	Existing C ant miner	New Pheromone Updator
Heart	6.31	6.5
Hepatitis	5.02	5.25
Ionosphere	6	6
WDBC	5.46	5.833

Table 4 : no of rules detail

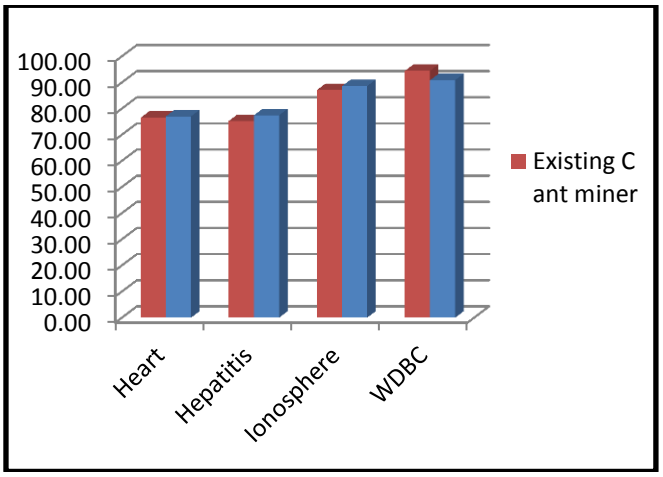

Figure. 1: Accuracy Comparison of Existing System and New Pheromone Updater

All rights reserved by www.ijsrd.com

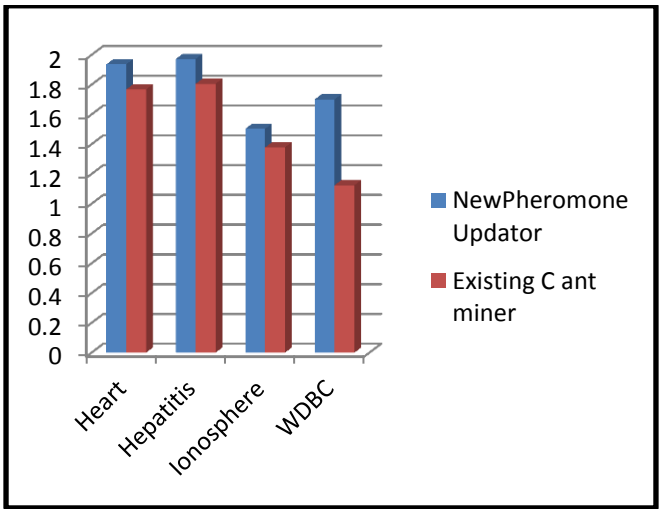

Figure. 2: Terms per Rule : Comparison of Existing System and New Pheromone Updater

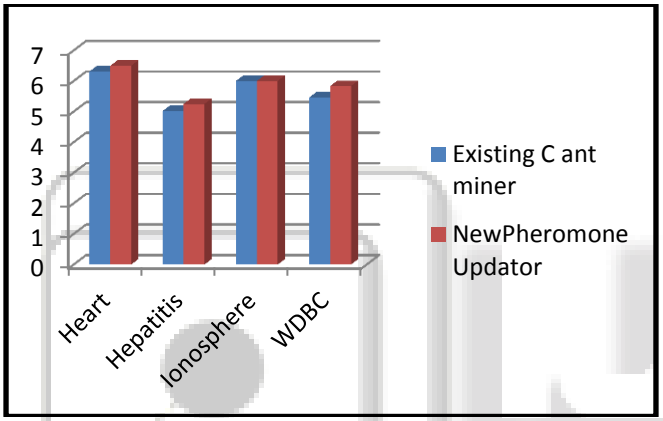

Figure. 3 No. of Rule generated : Comparison of Existing System and New Pheromone Updater

VI. CONCLUSION

Data mining predict future trends and behaviors, which helps to make proactive, knowledge-driven decisions .One of the data mining tasks is the classification rules extraction from databases helps in achieving this. Ant Miner is a technique that successfully incorporates swarm intelligence in data mining. The main objective of research is related to improve accuracy and generate better classification rules. For this we have selected the pheromone update method which helps to fulfill the task. By the change done in method of pheromone update as discussed in previous chapter Gain in accuracy of the solution is acquired by variation of 2-3%

REFERENCES

[1] R.S.Parnelli, H.S.Lopes and A.A.Freitas, Data Mining with an Ant Colony Optimization Algorithm, IEEE Trans. OnEvolutinary Computation, special issue on Ant colony Algorithm, 6(4),pp 321-332,Aug 2002.

[2] Bo,L., Abbas, H.A, Kay ,B Classification Rule Discovery With an Ant Colony optimization.In : International Conference on Intelligent Agent Technology,2003.IAT 2003. IEEE October 13-1-2003 ,pp 83-88(2003).

[3] Fernando E.B. Otero, Alex A. Freitas, and Colin G. Johnson, cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes in Springer-Verlag Berlin, Heidelberg 2008.

[4] K .Thangavel.,P. Jaganathan,Rule Mining with a new Ant Colony optimization Algorithm Rules In:2007 IEEE International Conference on Granular Computing ,pp 135-140(2007).

[5] Bo Liu , Hussein A.Abbass, Bob McKay, Density_based Heuristic for Rule Discovery with Ant_Miner, The 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System,2002, page 180-184.

[6] Fayyad, Usama; Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996). "From Data Mining to Knowledge Discovery in Databases"Retrieved 2008-12-17.

[7] S. Sumathi, S.N. Sivanandam ,Introduction to Data Mining and its Applications, Studies in Computational Intelligence, Volume 29, Springer Berlin Heidelberg New York

[8] Daniel t. Larose –DATA MINING METHODS AND MODELS, Department of Mathematical Sciences Central Connecticut State University

[9] Ghosh, L.C. Jain (Eds.),Evolutionary Computation in Data Mining , Studies in Fuzziness and Soft Computing,Volume 163, Springer Berlin Heidelberg New York

[10] Crina Grosan, Ajith Abraham and Monica Chis,Swarm Intelligence in Data Mining, Studies in Computational Intelligence (SCI) 34, Springer Berlin Heidelberg New York 1–20 (2006)

All rights reserved by www.ijsrd.com