Classification Rule Discovery with Ant Colony Optimization

***Bo Liu*[1], *Hussein A. Abbass*[2] *and Bob McKay*[2]**

[1]*College of Computer and Information Engineering, Guangxi University,Nanning, China,530004*

[2]*School of Computer Science, University of New South wales, Australia, ACT 2600*

ddxllb@yahoo.com.cn, *abbass@cs.adfa.edu.au*, *rim@cs.adfa.edu.au*

Abstract

Ant-based algorithms or ant colony optimization (ACO) algorithms have been applied successfully to combinatorial optimization problems. More recently, Parpinelli and colleagues applied ACO to data mining classification problems, where they introduced a classification algorithm called Ant_Miner. In this paper, we present an improvement to Ant_Miner (we call it Ant_Miner3). The proposed version was tested on two standard problems and performed better than the original Ant_Miner algorithm.

1.Introduction

Knowledge discovery in databases (KDD) is the process of extracting models and patterns from large databases. The term *data mining* (DM) is often used as a synonym for the KDD process, although strictly speaking it is just a step within KDD. DM refers to the process of applying the discovery algorithm to the data. In [5], KDD is defined as

"… the process of model abstraction from large databases and searching for valid, novel, and nontrivial patterns and symptoms within the abstracted model".

Rule Discovery is an important data mining task since it generates a set of symbolic rules that describe each class or category in a natural way. The human mind is able to understand rules better than any other data mining model. However, these rules need to be simple and comprehensive; otherwise, a human won't be able to comprehend them. Evolutionary algorithms have been widely used for rule discovery, a well known approach being learning classifier systems.

To our knowledge, Parpinelli, Lopes and Freitas [4] were the first to propose Ant Colony Optimization (ACO) for discovering classification rules, with the system *Ant-Miner*. They argue that an ant-based search is more flexible and robust than traditional approaches. Their method uses a heuristic value based on entropy measure.

In [9], we presented a modified version of Ant-Miner (i.e. Ant-Miner2), where the core computation heuristic value is based on a simple density estimation heuristic. In this paper, we present a further study and introduce another ant-based algorithm, which uses a different pheromone updating strategy and state transition rule. By comparison with the work of Parpinelli et al, our method can improve the accuracy of rule lists.

The remainder of the paper is organized as follow. In section 1, we present the basic idea of the ant colony system. In section 2, the Ant_Miner algorithm (Rafael S.Parpinelli et al, 2000) is introduced. In section 3, the density based Ant_miner2 is explained. In section 4, our further improved method (i.e.Ant_Miner3) is shown. Then the computational results are reported in section 5. Finally, we conclude with general remarks on this work and further directions for future research.

2. Ant Colony System (ACS) and Ant_Miner

Ant Colony Optimization (ACO) [2] is a branch of a newly developed form of artificial intelligence called *swarm intelligence*. Swarm intelligence is a field which studies "the emergent collective intelligence of groups of simple agents" [1]. In groups of insects, which live in colonies, such as ants and bees, an individual can only do simple tasks on its own, while the colony's cooperative work is the main reason determining the intelligent behavior it shows. Most real ants are blind. However, each ant while it is walking, deposits a chemical substance on the ground called pheromone [2]. Pheromone encourages the following ants to stay close to previous moves. The pheromone evaporates over time to allow

```
Training set = all training cases;
WHILE (No. of uncovered cases in the Training set > max_uncovered_cases)
    i=0;
    REPEAT
        i=i+1;
        Ant_i incrementally constructs a classification rule;
        Prune the just constructed rule;
        Update the pheromone of the trail followed by Ant_i;
    UNTIL (i ≥ No_of_Ants) or (Ant_i constructed the same rule as the
    previous No_Rules_Converg-1 Ants)
    Select the best rule among all constructed rules;
    Remove the cases correctly covered by the selected rule from the training
    set;
END WHILE
```

Figure 1. Overview of Ant-Miner (Parepinelli et al., 2002)

search exploration. In a number of experiments presented by [3], the complex behavior of the ants' colony is illustrated. For example, a set of ants built a path to some food. An obstacle with two ends was then placed in their way such that one end of the obstacle was more distant than the other. In the beginning, equal numbers of ants spread around the two ends of the obstacle. Since all ants have almost the same speed, the ants going around the nearer end of the obstacle return before the ants going around the farther end (differential path effect). With time, the amount of pheromone the ants deposit increases more rapidly on the shorter path, and so more ants prefer this path. This positive effect is called autocatalysis. The difference between the two paths is called the preferential path effect; it is the result of the differential deposition of pheromone between the two sides of the obstacle, since the ants following the shorter path will make more visits to the source than those following the longer path. Because of pheromone evaporation, pheromone on the longer path vanishes with time.

The goal of Ant-Miner is to extract classification rules from data (Parpinelli et al., 2002). The algorithm is presented in Figure 1.

2.1. Pheromone Initialization

All cells in the pheromone table are initialized equally to the following value:

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

where:

- a is the total number of attributes;
- b_i is the number of values in the domain of attribute i.

2.2. Rule Construction

Each rule in Ant-Miner contains a condition part as the antecedent and a predicted class. The condition part is a conjunction of attribute-operator-value tuples. The operator used in all experiments is "=" since in Ant-Miner2, just as in Ant-Miner, all attributes are assumed to be categorical. Let us assume a rule condition such as $term_{ij}$ ☒ $A_i=V_{ij}$, where A_i is the ith attribute and V_{ij} is the jth value in the domain of A_i. The probability, that this condition is added to the current partial rule that the ant is constructing, is given by the following Equation:

$$P_{ij}(t)=\frac{\tau_{ij}(t)\cdot\eta_{ij}}{\sum_{i}^{a}\sum_{j}^{b_i}\tau_{ij}(t)¿\eta_{ij}}, \forall i\in I \qquad (2)$$

Where:

• η_{ij} is a problem-dependent heuristic value for term$_{ij}$

• a is the total number of attributes

• b_i is the total number of values in the domain of attribute i

• I is the set of attributes that are not yet used by the ant

2.3. Heuristic Value

In traditional ACO, a heuristic value is usually used in conjunction with the pheromone value to decide on the transitions to be made. In Ant-Miner, the heuristic value is taken to be an information theoretic measure for the quality of the term to be added to the rule. The quality here is measured in terms of the entropy for preferring this term to the others, and is given by the following equations:

$$\eta_{ij} = \frac{\log_2(k) - InfoT_{ij}}{\sum_{i}^{a} \sum_{j}^{b_i} \log_2(k) - InfoT_{ij}} \qquad (3)$$

$$InfoT_{ij} = -\sum_{w=1}^{k} \left[\frac{freqT_{ij}^{w}}{|T_{ij}|} \right] * \log_2 \left[\frac{freqT_{ij}^{w}}{|T_{ij}|} \right] \qquad (4)$$

where:

• k is the number of class

• $|T_{ij}|$ is the total number of cases in partition T_{ij} (partition containing the cases where attribute A_i has value V_{ij})

• $freqT^{w}_{ij}$ is the number of cases in partition T_{ij} with class w

• a is the total number of attributes

• b_i is the number of values in the domain of attribute i

The higher the value of infoT$_{ij}$, the less likely that the ant will choose term$_{ij}$ to add to its partial rule.

2.4. Rule Pruning

Immediately after the ant completes the construction of a rule, rule pruning is undertaken to increase the comprehensibility and accuracy of the rule. After the pruning step, the rule may be assigned a different predicted class based on the majority class in the cases covered by the rule antecedent. The rule pruning procedure iteratively removes the term whose removal will cause a maximum increase in the quality of the rule.

• $\tau_{ij}(t)$ is the amount of pheromone currently available (at time t) on the connection between attribute i and value j

The quality of a rule is measured using the following equation

$$Q = \left(\frac{TruePos}{TruePos + FalseNeg} \right) \times \left(\frac{TrueNeg}{FalsePos + TrueNeg} \right)$$
(5)

Where:

• TruePos is the number of cases covered by the rule and having the same class as that being predicted by the rule

• FalsePos is the number of cases covered by the rule and having a different class from that being predicted by the rule

• FalseNeg is the number of cases that are not covered by the rule while having the class predicted by the rule

• TrueNeg is the number of cases that are not covered by the rule and having a different class from the class predicted by the rule.

2.5. Pheromone Update Rule

After each ant completes the construction of its rule, pheromone updating is carried out as follows:

$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t).Q$, $\forall$ term$_{ij}$ ¿ the rule
(6)

To simulate the phenomenon of pheromone evaporation in real ant colony systems, the amount of pheromone associated with each term$_{ij}$ which does not occur in the constructed rule must be decreased,. The reduction of pheromone of an unused term is performed by dividing the value of each τ_{ij} by the summation of all τ_{ij}.

3.Density based Ant_Miner2

In [9], we proposed an easily computable density estimation equation (7) instead of equation (3). It derived from the view that the ACO algorithm does not need accurate information in its heuristic value, since the pheromone should compensate for small potential errors in the heuristic values. In other words, a simpler heuristic value may do the job as well as a complex one. This simple heuristic produced equivalent results to the entropy function.

$$\eta_{ij} = \frac{majority_{classT_{ij}}}{|T_{ij}|} \qquad (7)$$

where: majority_classT_{ij} is the majority class in partition T_{ij}

4.Proposed System (Ant_Miner3)

It is reasonable that ants select terms in according to equation (1); in other words, determined by pheromone amount and heuristic function η which measures the predictive power of a term. But in the above methods, the pheromone of each term is changed after an ant constructs a rule, while η is always the same, so the next ant tends to choose terms used in the previous rule, whose pheromone is increased, and is unlikely choose unused terms, whose pheromone is decreased. Consequently, the ants converge to a single constructed rule too quickly. This leads to a failure to produce alternative potential rules.

In [9], we exemplified that Ant-Miner2 is computationally less expensive than the original Ant-Miner1, since it is based on simple division instead of the logarithm as in Ant-Miner. To be more precise, each heuristic value in Ant-Miner1 requires 2 divisions, 1 multiplication and 1 calculation of the logarithm, whereas Ant-Miner2 requires a single division. This saving in computational time did not change the accuracy of the method and did not require additional iterations.

In the following, we propose a new pheromone updating method and a new state transition rule to increase the accuracy of classification by ACO.

4.1. Our Pheromone Update Method

After an ant constructs a rule, the amount of pheromone associated with each term that occurs in the constructed rule is updated by equation (8), and the pheromone of unused terms is updated by normalization.

Note that Q varies in the range [0, 1]. The higher Q is, the larger the amount of pheromone associated with each used term. On the other hand, if Q is very small (close to zero), the pheromone level associated with each used term will decrease.

$$\tau_{ij}(t)=(1-\rho)\cdot\tau_{ij}(t-1)+\left(1-\frac{1}{1+Q}\right)\cdot\tau_{ij}(t-1) \tag{8}$$

where:

- ρ is the pheromone evaporation rate.
- Q is quality of the constructed rule.

ρ is the pheromone evaporation rate, which controls how fast the old path evaporates. This parameter controls the influence of the history on the current pheromone trail [6]. In our method, a large value of ρ indicates a fast evaporation and vice versa. We fix it at 0.1 in our experiments.

4.2.Choice of Transition

Ants can be regarded as cooperative agents in an ant colony system. These intelligent agents inhabit an environment without global knowledge, but they could benefit form the update of pheromones [7]. Pheromones placed on the edges in ACS play the role of a distributed long-term memory. This memory is not stored within the individual ants, but is distributed on the edges of the router, which allows an indirect form of communication. This benefits exploitation of prior knowledge. But it increases the probability of choosing terms belonging to the previously discovered rules according to equation (2), thus inhibiting the ants from performing biased exploration. In order to enhance the role of biased exploration, we apply the transition rule shown in figure 3 in choosing term$_{ij}$ where:

- q1 and q2 are random numbers;
- φ is a parameter in [0, 1] ;
- J_i is the number of i-th attribute values;
- P_{ij} is possibility calculated according to equation (2).

Therefore, the result depends not only on the heuristic functions η_{ij} and pheromone τ_{ij}, but also on random number, which increases the opportunity to choose terms unused in previously constructed rules. More precisely, q1≥φ corresponds to an exploitation of the knowledge available about the problem, whereas q1≤φ favors more exploration. Φ is tunable for controlling exploration. In our experiments, it is set to 0.4.

```
If  $q1 \le \varphi$
  Loop
    If  $q2 \le \sum_{j \in J_i} P_{ij}$
    Then  choose term$_{ij}$
  Endloop
Else
    Choose term$_{ij}$ with max $P_{ij}$
```

5. Ex

In the experiment, we used two data sets from the UCI data set repository[8]. The first is the Wisconsin breast cancer database, which contains 699 instances, 9

Table 1. Accuracy Rate of Different Groups (%)

	Breast Cancer		Tic_tac_toe	
Running Times	Ant_Miner1	Ant_Miner3	Ant_Miner1	Ant_Miner3
1	92.05	94.32	71.28	82.97
2	93.15	93.15	73.40	72.34
3	91.67	91.67	67.37	78.94
4	95.59	97.06	71.58	80.00
5	88.41	92.75	68.42	72.63
6	94.20	95.65	75.79	80.00
7	90.77	93.84	74.74	81.05
8	96.55	96.55	65.26	74.74
9	91.04	92.54	73.68	75.79
10	92.86	95.71	68.42	67.37

integer-valued attributes and 2 classes (malignant and benign). The second is the Tic_tac_toe endgame database, which contains 958 instances, 9 numerate-valued attributes and 2 classes (won and lose). We evaluate performance of the proposed method by comparison with Parpinelli's Ant_Miner1 algorithm. Each Database is divided into ten partitions. Each method is run ten times. Each time, a different partition is used as the test set and the other nine partitions are used as the training set.

We use the rule list produced by a training set to predict the class of each case in the test set, the accuracy rate being calculated according to equation (9) (where the meanings of TruePos, Trueneg, FalseNeg, FalsePos are as in equation (5)). Every rule list includes a default rule which has no condition and as its class, the majority class in the set of training cases, so that we can apply the default rule if none of the rules in the list covers test case.

Table 1 shows accuracy rates for the rule sets produced by Ant_miner1 and Ant_Miner3 running ten times on the two datasets. The mean accuracy rate and mean number of rule sets by both methods are reported in Table 2. It can be concluded that Ant_Miner3 discovers somewhat more rules than Ant_Miner1, but the mean accuracy of the rule sets discovered by Ant_Miner3 is higher than Ant_Miner1. We also conduct one-tail student's t-tests. The significant level is 0.04 for the Breast Cancer Database and 0.004 for the Tic-tac-toe. Therefore the difference of two methods is statistically significant. This shows that if ants explore more different paths, then there is a higher probability that one of them will find an improved solution rather than the case in which they all converge to the same tour.

Although Ant_Miner3 requires marginally more ants to find a solution, the density based heuristic computational method compensates for ants searching time. Therefore, Ant_miner1 and Ant_miner3 consume almost the same running time.

6.Conclusion

There are a number of traditional ways to discover rules in databases, such as decision-tree induction. In the literature [4], it is exemplified that Ant-Miner1 produces higher accuracy rate and fewer rules than decision-tree induction (C4.5). In this paper, a new method based on a variant of Ant_Miner1 is proposed. We compare the results of both methods and found that our method features a higher accuracy rate than Ant_Miner1. The main contributions of the work are the following:

1.Our method considers random factors when constructing a rule, and so comprises both exploitation and exploration in its operation. This more accurately models the behavior of real ants, but also, because it leads to a greater diversity of path choices, assists in finding the optimal rule.

2.A different strategy for controlling the influence of pheromone values was studied. We proposed a pheromone update rule which can cause future ants to make better decisions, i.e. improving the quality of a rule and the accuracy of rule sets.

$$\text{Accuracy Rate}=\frac{TruePos+TrueNeg}{TruePos+FalseNeg+FalsePos+TrueNeg} \quad (9)$$

Table 2. Mean accuracy rate and mean number of rule lists

	Breast Cancer		Tic_tac_toe	
Valuation item	Ant_Miner1	Ant_Miner3	Ant_Miner1	Ant_Miner3
Accuracy rate(%)	92.63	94.32	70.99	76.58
No_of_rules	10.1	13.2	16.5	18.58

Applying ACO in data mining is still in its early stages. In the future, we would like to improve time efficiency further and tune some system parameters, so that better results might be found and the ACO method can be applied in real-world large scale databases.

References

[1] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. New York: Oxford University Press.

[2] Dorigo, M., & Caro, G. D. (1999). Ant Algorithms for Discrete Optimization. Artificial Life, 5(3), 137-172.

[3] Dorigo, M., & Maniezzo, V. (1996). The ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, 26(1), 1-13.

[4] Parepinelli, R. S., Lopes, H. S., & Freitas, A. (2002). An Ant Colony Algorithm for Classification Rule Discovery. In H. A. a. R. S. a. C. Newton (Ed.), Data Mining: Heuristic Approach: Idea Group Publishing.

[5] Sarker, R., Abbass, H., & Newton, C. (2002). Introducing data mining and knowledge discovery. In R. sarker & H. Abbass & C. Newton (Eds.), Heuristics and Optimisation for Knowledge Discovery (pp. 1-23): Idea Group Publishing.

[6] Luk Schoofs, Bart Naudts, Ant Colonies are Good at Solving Constraint Satisfaction Problems, Proceedings of the 2000 Congress on Evolutionary Computation, Volume:2, page 1190-1195.

[7] Ruoying Sun, Shoji Tatsumi, Gang Zhao, Multiagent Reinforcement Learning Method with An Improved Ant Colony System, Proceedings of the 2001 IEEE International Conference on Systems, Man and Cybernetics, Volume:3, 2001, page 1612-1617.

[8] Hettich, S. and Bay, S.D.(1999). The UCI KDD Archive, Retrieved September, 2002, from the World Wide Web: http://kdd.ics.uci.edu.

[9] Bo Liu , Hussien 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.