Ant ColonyAlgorithms

for Data Classification

Alex A. Freitas

University of Kent, Canterbury, UK

Rafael S. Parpinelli

UDESC, Joinville, Brazil

Heitor S. Lopes

UTFPR, Curitiba, Brazil

INTRODUCTION

Ant Colony Optimization (ACO) is a relatively new computational intelligence paradigm inspired by the behaviour of natural ants (Dorigo & Stutzle, 2004). Ants often find the shortest path between a food source and the nest of the colony without using visual information. In order to exchange information about which path should be followed, ants communicate with each other by means of a chemical substance called pheromone. As ants move, a certain amount of pheromone is dropped on the ground, creating a pheromone trail. The more ants

follow a given trail, the more attractive that trail becomes to be followed by other ants. This process involves a loop of positive feedback, in which the probability that an ant chooses a path is proportional to the number of ants that have already passed by that path.

Hence, individual ants, following very simple rules, interact to produce an intelligent behaviour at the higher level of the ant colony. In other words, intelligence is an emergent phenomenon.

In this article we present an overview of Ant-Miner, an ACO algorithm for discovering classification rules in data mining (Parpinelli, Lopes & Freitas, 2002a; Parpinelli, Lopes & Freitas, 2002b), as well as a review of several Ant-Miner variations and related ACO algorithms.

All the algorithms reviewed in this article address the classification task of data mining. In this task each case (record) of the data being mined consists of two parts: a goal attribute, whose value is to be predicted, and a set of predictor attributes. The aim is to predict the value of the goal attribute for a case, given the values of the predictor attributes for that case (Fayyad, Piatetsky-Shapiro & Smyth, 1996).

BACKGROUND

An ACO algorithm is essentially a computational system inspired by the behavior of natural ants and designed to solve real-world optimization problems. The basic ideas of ACO algorithms are as follows (Dorigo & Stutzle, 2004):

- Each ant incrementally constructs a candidate solution to a given optimization problem. That candidate solution is associated with a path in graph representing the search space;
- When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution;
- At each step during the incremental construction of a candidate solution, an ant typically has to choose which solution component should be added to the current partial solution (i.e., how to extend the current path), among several solution components. In general the probability of a given component being chosen is proportional to the product of two terms: (a) the amount of pheromone associated with that component; and (b) the value of a (problem-dependent) heuristic function for that component.

As a result, due to a continuous increase of the pheromone associated with the components of the best candidate solutions considered by the algorithm, the ants usually converge to the optimum or near-optimum solution for the target problem.

The motivation for applying ACO algorithms to the discovery of classification rules and related data mining tasks is as follows. Many projects in the field of data mining proposed deterministic rule induction algorithms. These algorithms typically are greedy, and so they are susceptible to find only locally-optimal (rather than globally optimal) classification rules. By contrast, ACO algorithms try to mitigate this drawback using a combination of two basic ideas. First, these algorithms have a stochastic aspect, which helps them to explore a larger area of the search space. Second, they use an iterative adaptation procedure

based on positive feedback (the gradual increase of the pheromone associated with the best solution components) to continuously improve candidate rules. Combining these basic ideas, in general ACO algorithms perform a more global search in the space of candidate rules than typical deterministic rule induction algorithms, which makes the former an interesting alternative to be considered in rule induction.

The first ACO algorithm proposed for discovering classification rules was Ant-Miner (Parpinelli, Lopes & Freitas, 2002a). In Ant-Miner each artificial path constructed by an ant represents a candidate classification rule of the form:

IF <*term1* AND *term2* AND … > THEN < *class* >.

Each term is a triple <*attribute, operator, value*>, where *value* is one of the values belonging to the domain of *attribute*. An example of a term is: <*Sex* = *female*>. *Class* is the value of the goal attribute predicted by the rule for any case that satisfies all the terms of the rule antecedent. An example of a rule is:

IF <*Salary* = *high*> AND <*Mortgage* = *No*> THEN <*Credit* = *good*>.

In the original version of Ant-Miner the *operator* is always "=", so that Ant-Miner can cope only with categorical (discrete) attributes. Continuous attributes would have to be discretized in a preprocessing step.

The pseudocode of Ant-Miner is described, at a very high level of abstraction, in Algorithm 1. Ant-Miner starts by initializing the training set to the set of all training cases, and initializing the discovered rule list to an empty list. Then it performs an outer loop where each iteration discovers a classification rule.

The first step of this outer loop is to initialize all trails with the same amount of pheromone, which means that all terms have the same probability of being chosen – by an ant – to incrementally construct a rule. This is done by an inner loop, consisting of three steps. First, an ant starts with an empty rule and incrementally constructs a classification rule by adding one term at a time to the current rule. In this step a $term_{ij}$ – representing a triple $<Attribute_i = Value_j>$ – is chosen to be added to the current rule with probability proportional to the product of $\eta_{ij} \times \tau_{ij}(t)$, where η_{ij} is the value of a problem-dependent heuristic function for $term_{ij}$ and $\tau_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration (time index) t. More precisely, η_{ij} is essentially the information gain associated with $term_{ij}$ – see (Cover & Thomas, 1991) for a comprehensive discussion on information gain. The higher the value of η_{ij} the more relevant for classification $term_{ij}$ is and so the higher its probability of being chosen. $\tau_{ij}(t)$ corresponds to the amount of pheromone currently available in the position i,j of the trail 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 positions ("terms") visited ("used") by the ant. Therefore, as time goes by, the best trail positions to be followed – i.e., the best terms to be added to a rule – will have greater and greater amounts of pheromone, increasing their probability of being chosen.

The second step of the inner loop consists of pruning the just-constructed rule, i.e., removing irrelevant terms. This is done by using the same rule-quality measure used to update the pheromones of the trails, as defined later. In essence,

a term is removed from a rule if this operation does not decrease the quality of the rule.

```
TrainingSet = {all training cases};
DiscoveredRuleList = [];   /* initialized with empty list */
REPEAT
   Initialize all trails with the same amount of pheromone;
   REPEAT
      An ant incrementally constructs a classification rule;
      Prune the just-constructed rule;
      Update the pheromone of all trails;
   UNTIL (stopping criteria)
   Choose the best rule out of all constructed rules;
   Add the chosen rule to DiscoveredRuleList;
   TrainingSet = TrainingSet – {cases correctly covered by the chosen rule};
UNTIL (stopping criteria)
```

Algorithm 1: High-level pseudocode of Ant-Miner

The third step of the inner loop consists of updating the pheromone of all trails by increasing the pheromone in the trail followed by the ant, proportionally to the quality of the rule. In other words, the higher the quality of the rule, the higher the increase in the pheromone of the terms occurring in the rule antecedent. The quality (Q) of a rule is measured by the equation:

$$Q = \text{Sensitivity} \times \text{Specificity},$$

where $Sensitivity = TP / (TP + FN)$ and $Specificity = TN / (TN + FP)$. The acronyms TP, FN, TN and FP stand for the number of true positives, false negatives, true negatives and false positives, respectively (Parpinelli, Lopes, Freitas, 2002a).

The inner loop is performed until some stopping criterion(a) is(are) satisfied, e.g., until a maximum number of candidate rules has been constructed.

Once the inner loop is over, the algorithm chooses the highest-quality rule out of all the rules constructed by all the ants in the inner loop, and then it adds the chosen rule to the discovered rule list. Next, the algorithm removes from the training set all the cases correctly covered by the rule, i.e., all cases that satisfy the rule antecedent and have the same class as predicted by the rule consequent. Hence, the next iteration of the outer loop starts with a smaller training set, consisting only of cases which have not been correctly covered by any rule discovered in previous iterations. The outer loop is performed until some stopping criterion is satisfied, e.g., until the number of uncovered cases is smaller than a user-specified threshold.

Hence, the output of Ant-Miner is the list of classification rules contained in discovered rule list.

(Parpinelli, Lopes & Freitas, 2002a; Parpinelli, Lopes & Freitas, 2002b) have performed computational experiments comparing Ant-Miner with two well-known rule induction algorithms, namely CN2 (Clark & Niblett, 1989) and C4.5 (Quinlan, 1993) in several public-domain data sets. In a nutshell, the results showed that Ant-Miner is competitive with both C4.5 and CN2 concerning predictive accuracy on the test set. However, Ant-Miner discovered rule lists

much simpler (i.e., smaller) than the rule sets discovered by C4.5 and the rule lists discovered by CN2. This is an advantage in the context of data mining, where discovered knowledge is supposed to be comprehensible to the user – in order to support the user's intelligent decision making (Fayyad, Piatetsky-Shapiro & Smyth, 1996).

ANT-MINER VARIATIONS AND RELATED ALGORITHMS

(a) Fixing in Advance the Class Predicted by a Rule

In Ant-Miner, each ant first constructs a rule antecedent. Next, the majority class among all cases covered by the rule is assigned to the rule consequent. The fact that the class predicted by a rule is not known during rule construction has two important consequences. First, the heuristic function is based on reducing the entropy associated with the entire class distribution, rather than using a heuristic function specific to the class to be predicted by the rule. Second, the pheromone associated with each term represents the relevance of that term with respect to the overall discrimination between all classes, rather than with respect to a specific class.

In order to make the heuristic function and pheromone values have a more focused relevance, variations of Ant-Miner have been proposed where all the ants in the population construct rules predicting the same class, so that the class to be assigned to a rule is known during rule construction (Chen, Chen & He, 2006;

Galea & Shen, 2006; Martens et al., 2006; Smaldon & Freitas, 2006). This naturally leads to new heuristic functions and new pheromone update methods, as discussed in items (b) and (e).

(b) New Class-Specific Heuristic Functions

Several Ant-Miner variations have replaced the entropy reduction heuristic function by a simpler heuristic function whose value is specific for each class (Chen, Chen & He, 2006; Liu, Abbass & McKay, 2004; Martens et al., 2006; Oakes, 2004; Smaldon & Freitas, 2006; Wang & Feng, 2004). Typically, in these Ant-Miner variations, the value of the heuristic function for a $term_{ij}$ is based on the relatively frequency of the class predicted by the rule among all the cases that have the $term_{ij}$. This modification is particularly recommended when the class predicted by a rule is fixed in advance before rule construction, unlike the situation in the original Ant-Miner – as discussed in item (a).

It has also been argued that a simpler heuristic function based on the relative frequency of the rule's predicted class has the advantage of being computationally more efficient without sacrificing the predictive accuracy of the discovered rules, since hopefully the use of pheromones should compensate for the less accurate estimate of the simpler heuristic function. Note, however, that there is no guarantee that the use of pheromones would completely compensate the use of a less effective heuristic function. A more important issue seems to be whether or not the rule's predicted class is known during rule construction, as discussed earlier. In addition, note that in Ant-Miner the heuristic function values are computed just once in the initialization of the algorithm, and in principle the

heuristic function values for all terms can be computed in linear time with respect to the number of cases and attributes (Parpinelli, Lopes & Freitas, 2002a). Hence, the time complexity of the heuristic function of Ant-Miner does not seem a significant problem in the context of the entire algorithm.

(c) Using the Pseudorandom Proportional Transition Rule

As explained earlier, Ant-Miner adds a $term_{ij}$ to a rule with a probability proportional to the product $\eta_{ij} \times \tau_{ij}(t)$. This kind of transition rule is called the random proportional transition rule (Dorigo & Stutzle, 2004). Several variants of Ant-Miner use instead a pseudorandom proportional transition rule (Chen, Chen & He, 2006; Liu, Abbass & McKay, 2004; Wang & Feng, 2004). In this transition rule, in essence, there is a probability q_0 that the term to be added to the current partial rule will be deterministically chosen as the $term_{ij}$ with the greatest value of the product of $\eta_{ij} \times \tau_{ij}(t)$; and there is a probability $1 - q_0$ that the term to be added to the rule will be probabilistically chosen by the random proportional rule. This transition rule has the advantage that it allows the user to have an explicit control over the exploitation vs. exploration trade-off (Dorigo & Stutzle, 2004), which of course comes with the need to choose a good value for the parameter q_0 – normally chosen in an empirical way.

(d) New Rule Quality Measures

Ant-Miner's rule quality is based on the product of sensitivity and specificity. (Chen, Chen & He, 2006) and (Martens et al., 2006) proposed to replace Ant-

Miner's rule quality measure by measures that are essentially based on the confidence and coverage of a rule. Coverage is equivalent to sensitivity, so the main idea introduced in these Ant-Miner variations is to replace specificity by confidence in the rule quality measure. Given the importance of rule quality – on which pheromone updating is based – it would be interesting to perform extensive experiments comparing the effectiveness of these two kinds of rule quality measures, something missing in the literature.

(e) New Pheromone Updating Procedures

Several variants of Ant-Miner use an explicit pheromone evaporation rate – this is a predefined parameter in (Liu, Abbass & McKay, 2004), (Martens et al., 2006) and a self-adaptive parameter in (Wang & Feng, 2004). Note that this new parameter is not necessary in the original Ant-Miner, where pheromone evaporation is implicitly performed by normalizing all pheromone values after increasing the pheromone of the terms used in the just-constructed rule.

In addition, (Liu, Abbass & McKay, 2004), (Wang & Feng, 2004) and (Smaldon & Freitas, 2006) proposed different equations or procedures to update pheromone as a function of rule quality. These new approaches address the issue that Ant-Miner's original equation for pheromone updating does not work well when the quality of a rule is close to zero. Note, however, that in the original Ant-Miner the value of the rule quality measure will be close to zero only in rare situations. This is due to the fact that the best possible class for the current rule is always chosen to be added to the rule consequent after the rule antecedent has been fully constructed, which should result in a rule of at least reasonable quality

in most cases. On the other hand, in Ant-Miner variants where the class of all constructed rules is predetermined – see the discussion in item (a) – rules with very low quality will be less rare, especially in early iterations of the algorithm, before the pheromone of good terms has been significantly increased. In such Ant-Miner variants it is indeed important to change the original Ant-Miner's equation to update pheromone in order to cope better with low-quality rules.

(f) Coping with Categorical Attributes Having Ordered Values

Ant-Miner copes only with categorical (nominal or discrete) attributes. Hence, continuous (real-valued) attributes have to be discretized in a preprocessing step. Some categorical attributes have unordered values (e.g., the attribute "gender" can take the values "male" or "female"), whilst other categorical attributes have ordered values (e.g., the attribute "number of children" could take the values "0", "1", "2", "3" or "more than 3"). Ant-Miner does not recognize this distinction. It produces only terms (rule conditions) of the form "attribute = value", using the "=" operator.

Both (Oakes, 2004) and (Martens et al., 2006) proposed Ant-Miner variants that can cope with ordered values of categorical (but not continuous) attributes. These variants can discover rules containing terms of the form "attribute op value" where *op* can be a relational comparison operator such as "<", ">", "≤" or "≥". This allows the discovery of simpler (smaller) rule sets. For instance, a single condition like "number of children < 3" is shorter than the disjunction of

conditions: "number of children = 0" or "number of children = 1" or "number of children = 2".

(g) Dropping the Rule Pruning Procedure

(Martens et al., 2006) proposed an Ant-Miner variant (Ant-Miner+) which does not perform rule pruning. In order to mitigate the need for pruning, the domain of each attribute was extended with a dummy value, hereafter called the "any" value. Hence, for each attribute i with k values, there is a set of k+1 terms that can be chosen to be added to the current partial rule, where the $term_{i,k+1}$ is the "any" value for attribute i. Adding an "any" value term to a rule means that the attribute in question can have any value when the rule is evaluated, which effectively means that attribute is not present in the rule antecedent. The authors argue that, using this approach, pruning is superfluous because the dummy values already lead to shorter rules. The removal of pruning has the advantage of making Ant-Miner+ significantly faster than Ant-Miner, since rule pruning tends to be the most computational expensive and least scalable part of Ant-Miner (Parpinelli, Lopes & Freitas, 2002a).

On the other hand, from the point of view of predictive accuracy of the discovered rules, it is not very clear if rule pruning is really superfluous due to the use of dummy "any" values to make rules shorter. In the original Ant-Miner rule pruning is performed by a deterministic procedure that iteratively removes one term at a time from the rule as long as rule quality is improved. This kind of local search not only reduces the discovered rules' size, but also tends to improve the predictive accuracy of those rules (Parpinelli, Lopes & Freitas, 2002a). It

would be interesting to evaluate if, despite the relatively short size of the rules discovered by Ant-Miner+, the predictive accuracy of those rules could be increased by the use of a deterministic rule pruning procedure driven by rule quality.

(h) Discovering Fuzzy Classification Rules

(Galea & Shen, 2006) proposed a new ACO algorithm – called FRANTIC-SRL (Fuzzy Rules from ANT-Inspired Computation – Simultaneous Rule Learning) – that discovers fuzzy classification rules, rather than the crisp rules discovered by Ant-Miner. FRANTIC-SRL maintains multiple populations of ants, where each population is in charge of discovering rules predicting a different class. Hence, the class predicted by a rule is fixed in advance for all ants in each of the populations – see item (a).

Each ant constructs a fuzzy rule by adding one linguistic term (a fuzzy condition) at a time to the current partial rule. This involved replacing the heuristic function of Ant-Miner by a function based on fuzzy systems' theory. The new function is class-specific, i.e., the heuristic function of a term depends on the class predicted by the rule – see item (b). At each iteration, instead of evaluating each candidate rule separately, FRANTIC-SRL evaluates each candidate rule set containing one rule from each of the ant colony populations – i.e., each candidate rule set containing one rule for each class. Hence, at each iteration the number of evaluated rule sets is $numAnts^{numClasses}$, where *numAnts* is the number of ants in each population and *numClasses* is the number of classes. To speed up this step, the number of evaluated rule sets could be reduced by considering only

combinations of the best rules (rather than all rules) from each population, as pointed out by the authors.

(i) Discovering Rules for Multi-Label Classification

Multi-label classification involves the simultaneous prediction of the value of two or more class attributes, rather than just one class attribute as in conventional classification. (Chan & Freitas, 2006) proposed a major extension of Ant-Miner to cope with multi-label classification, called MuLAM (Multi-Label Ant-Miner). In MuLAM, each ant constructs a set of rules – rather than a single rule – where different rules predict different class attributes. A rule can predict a single class attribute or multiple class attributes, depending on which option will lead to better rules for the data being mined. MuLAM uses a pheromone matrix for each of the class attributes. Hence, when a rule is built, the terms in its antecedent are used to update only the pheromone matrix(ces) of the class attribute(s) predicted by that rule.

FUTURE TRENDS

In the last few years there has been an increasing interest in specialized types of classification problems which tend to be particularly challenging, such as multi-label classification – where a single rule can predict multiple classes; hierarchical classification – where the classes to be predicted are arranged in a hierarchy; and the discovery of fuzzy classification rules. Recently, the first variations of Ant-

Miner for some of these challenging classification problems have been proposed, as discussed earlier (see also Conclusions), and it seems that a future trend could be the development of more sophisticated Ant-Miner variations or related algorithms for these types of problems.

CONCLUSIONS

This article has reviewed Ant-Miner, the first ant colony algorithm for discovering classification rules in data mining, as well as a number of variations of the original algorithm. Some of the proposed variations were relatively simple, in the sense that they did not affect the type of classification rules discovered by the algorithm. However, some variations were proposed to cope with attributes having ordered categorical values, somewhat improving the flexibility of the rule representation language. Even more significant variations were also proposed. In particular, Ant-Miner variations have been proposed to discover multi-label classification rules and to discover fuzzy classification rules. The problem of extending Ant-Miner to cope with continuous attributes on-the-fly, during the run of the algorithm (rather than requiring continuous attributes to be discretized in a preprocessing phase) remains an open problem and an interesting future research direction.

REFERENCES

Chan, A. & Freitas, A.A. (2006). A new ant colony algorithm for multi-label classification with applications in bioinformatics. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2006)*, 27-34. San Francisco, CA: Morgan Kaufmann.

Chen, C; Chen, Y. & He, J. (2006). Neural network ensemble based ant colony classification rule mining. *Proc. First Int. Conf. Innovative Computing, Information and Control (ICICIC'06)*, 427-430.

Clark, P. & Niblett, T. (1989). The CN2 rule induction algorithm. *Machine Learning*, 3(4), 261-283.

Cover, T.M. & Thomas, J.A. (1991). *Elements of Information Theory*. New York: Wiley.

Dorigo, M. & Stutzle, T. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.

Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery: an overview. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (eds.) *Advances in* Knowledge *Discovery & Data Mining*, Cambridge, MA, USA: MIT Press, 1-34.

Galea, M. & Shen, Q. (2006). Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In: Agraham, A.; Grosan, C. and Ramos, V. (eds.) *Swarm Intelligence in Data Mining*, 75-99. Berlin: Springer.

Liu, B.; Abbass, H.A. & McKay, B. (2004). Classification rule discovery with ant colony optimization. *IEEE Computational Intelligence Bulletin, 3(1)*, 31-35.

Martens, D.; De Backer, M.; Haesen, R., Baesens, B. & Holvoet, T. (2006). Ants constructing rule-based classifiers. In: Agraham, A.; Grosan, C. and Ramos, V. (eds.) *Swarm Intelligence in Data Mining*, 21-43. Berlin: Springer.

Oakes, M.P. (2004). Ant colony optimisation for stylometry: the federalist papers. *Proc. Conf. on Recent Advances in Soft Computing (RASC-2004)*, 86-91.

Parpinelli, R.S., Lopes, H.S. & Freitas, A.A. (2002a). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation, Special Issue on Ant Colony Algorithms*, 6(4), 321-332.

Parpinelli, R.S., Lopes, H.S. & Freitas, A.A. (2002b). An ant colony algorithm for classification rule discovery. In: Abbass, H., Sarker, R., & Newton, C. (eds.). *Data Mining: a Heuristic Approach*, London: Idea Group Publishing, 191-208.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann.

Smaldon, J. & Freitas, A.A. (2006). A new version of the Ant-Miner algorithm discovering unordered rule sets. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2006)*, 43-50. San Francisco, CA: Morgan Kaufmann.

Wang, Z. & Feng, B. (2004). Classification rule mining with an improved ant colony algorithm. *AI 2004: Advances in Artificial Intelligence, LNAI 3339,* 357-367. Berlin, Springer.

KEY TERMS

Classification Rule: A rule of the form: IF (conditions) THEN (class), meaning that if a case (record) satisfies the rule conditions, it is predicted to have the class specified in the rule.

Data Mining: A research field where the goal is to discover accurate, comprensible knowledge (or patterns) in data.

Rule List: An order list of IF-THEN classification rules discovered by the algorithm during training. When the rules are applied to classify a testing case, they are applied in order, so that the first rule matching the testing case is used to classify that case.

Testing case: Each of the cases (records) of the test set.

Test Set: A set of cases unseen during the training of the algorithm. The test set is used to measure the predictive accuracy (generalisation ability) of the rules discovered during training.

Training case: Each of the cases (records) of the training set.

Training Set: A set of cases used by the algorithm to discover classification rules.