Ant Colony Algorithms for Data Classification

Alex A. Freitas
University of Kent, UK

Rafael S. Parpinelli
UDESC, Brazil

Heitor S. Lopes
UTFPR, Brazil

INTRODUCTION

Ant colony optimization (ACO) is a relatively new computational intelligence paradigm inspired by the behavior 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 that 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 behavior 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, 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 a 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.

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

Parpinelli et al. (2002a, 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 et al., 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, De Backer, M., Haesen, Baesens, & Holvoet, 2006; Smaldon & Freitas, 2006). This naturally leads to new heuristic functions and new pheromone update methods, as discussed in the next two subsections.

(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 et al., 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 relative 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 above.

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 et al., 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 instead use a pseudorandom proportional transition rule (Chen et al., 2006; Liu et al., 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 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 et al. (2006) and Martens et al. (2006) proposed to replace Ant-Miner's rule quality measure with