# Discovering Classification Rules for Email Spam Filtering with an Ant Colony Optimization Algorithm

El-Sayed M. El-Alfy, *Senior Member, IEEE*

***Abstract*—The cost estimates for receiving unsolicited commercial email messages, also known as spam, are threatening. Spam has serious negative impact on the usability of electronic mail and network resources. In addition, it provides a medium for distributing harmful code and/or offensive content. The work in this paper is motivated by the dramatic increase in the volume of spam traffic in recent years and the promising ability of ant colony optimization in data mining. Our goal is to develop an ant-colony based spam filter and to empirically evaluate its effectiveness in predicting spam messages. We also compare its performance to three other popular machine learning techniques: Multi-Layer Perceptron, Naïve Bayes and Ripper classifiers. The preliminary results show that the developed model can be a remarkable alternative tool in filtering spam; yielding better accuracy with considerably smaller rule sets which highlight the important features in identifying the email category.**

## I. Introduction

With the proliferation of electronic mail usage as a means for personal and business communication, the volume of unwanted messages that are received is growing as well. Due to its low cost for the senders and ease of deployment, several people and companies use it to quickly distribute unsolicited bulk messages, also called spam, to a large number of recipients. The reasons for sending spam vary and may include marketing of products and services (*e.g*. drugs, software, and health insurance), spreading rumors and other fraudulent advertisements (such as make-money fast), and distributing offensive content (such as adult material and pornographic images). Moreover, spam provides a medium for phishing attacks and distributing harmful content such as viruses, Trojan horses, worms and other malware. Spam has become a major threat for business users, network administrators and even ordinary users [1].

Due to the alarming increase of the spam volume and its serious impact, providing vigilantly spam fighters has recently attracted considerable attention. In addition to regulations and legislations, several technical solutions including commercial and open-source products have been proposed and deployed to alleviate this problem [2]. Installing anti-spam filters at the network gateway is among the most commonly used mechanism to block or quarantine spam messages as early as possible before they enter the users' mailboxes. Spam filtering methods fall into two broad categories: non-machine learning based and machine learning based. Most of the early anti-spam tools belonged to the former category; for instance using a blacklist of known spammers, a white list of safe senders, or a human-crafted list of keywords such as "Get Rich" either in the subject line or the message content [3]. However, these static lists are simple to be got around easily by spammers; for example by changing or spoofing the sender's address or domain each time. Spammers have also learnt to deliberately avoid/misspell words or forge the content to bypass the spam filters. These methods also require periodic manual update and the likelihood to filter out a legitimate message as spam is high which can be more serious than not filtering at all. According to the estimates by the British Computer Society (BCS), inaccurate anti-spam solutions may be responsible for wasting over five million working hours a year for users to check that legitimate messages were not mistakenly quarantined [4].

Unlike traditional techniques, machine learning based methods automatically analyze the content of received messages and build more robust models accordingly. Therefore, they can be more effective and dynamically updated to cope with spammers' tactics. Several machine-learning methods have been recently used for spam filtering, including support vector machines [5], memory-based learning [6], Ripper rule-based learning [7], neural networks [8], Bayesian classifiers [8, 9], fuzzy similarity [10], and abductive networks [11]. With the ever-increasing sophistication of spammers' software, it has become clear that no single technique could completely solve this problem. This has led some researchers to propose the application of a myriad of different techniques at several tiers, as each may excel in some prediction aspect [12].

Ant-colony optimization (ACO) has been successfully applied to a wide spectrum of challenging problems [13]. One of such problems, that has become incredibly an active research area, is data mining, which refers to the process of automated data analysis to detect relationships among data items and discover knowledge from the data that is accurate and comprehensible for the user. Learning classification rules from instances based on ant-colony optimization has been addressed in a number of papers [14-20]. However, to the best of our knowledge, evaluating its effectiveness in filtering spam is still unexplored. This study is motivated by the promising ability of ant-colony optimization in data

Manuscript received November 14, 2008. This work was supported in part by King Fahd University of Petroleum and Minerals, Saudi Arabia.

E. M. El-Alfy is with the King Fahd University of Petroleum and Minerals, College of Computer Sciences and Engineering, Dhahran 31261, Saudi Arabia (phone: +(966) 3-860-1930; fax: +(966) 3-860-2174; e-mail: alfy@ kfupm.edu.sa).

978-1-4244-2959-2/09/$25.00 © 

mining. Our goal in this paper is two folds. First we develop a spam filter methodology based on ant-colony optimization; we call it AntSFilter. Then, we investigate its effectiveness in discovering classification rules from an email corpus using a public domain dataset. We also compare its performance to some other state-of-the-art machine learning techniques, including Multi-Layer Perceptron (MLP) and Naïve Bayesian (NB) and Ripper classifiers.

The remainder of the paper is organized as follows. Section II provides a brief overview of ant-colony based data mining. Section III explains the procedure for constructing an ant-colony based spam filter. Section IV describes the dataset used to evaluate the effectiveness of the proposed approach. Section V presents the experimental results and compares the performance of the proposed approach with some other existing techniques. Finally, Section VI concludes the paper and pinpoints some directions for future work.

## II. Ant Colony Based Data Mining Overview

Ant colony optimization (ACO) is a widely recognized population-based meta-heuristic technique inspired by the natural foraging behavior of real ant colonies [13]. Due to the cooperative behavior of ants, they can learn the shortest path between their nest and the food source. Ants communicate indirectly by depositing a chemical substance called pheromone as they move. The pheromone level is decreased gradually on all paths as a result of evaporation. However, the amount of pheromone deposited by each ant is proportional to the quality of the path it follows. Thus shorter paths will have higher levels of pheromone and be more attractive to be followed by other ants; hence more pheromone is deposited. In artificial ants (agents), paths represent candidate solutions to the target problem.

Since its inauguration by Marco Dorigo in his Ph.D. thesis in 1992, ACO has been applied to a wide spectrum of practical problems in various disciplines. It has demonstrated robust and effective solutions for difficult combinatorial optimization problems. Some examples include scheduling, vehicle and network routing, quadratic assignment, multiple knapsack, timetabling, and traveling salesman problem. However, its application to the area of data mining was relatively recent. In 2001, Parpinelli *et al.* introduced an Ant-Miner algorithm that is based on ant-colony optimization for solving classification problems [14, 15]. They applied the proposed algorithm to four public domain medical datasets. They found that using Ant-Miner can give higher predictive accuracy for the considered cases than CN2, a well-known data mining algorithm for classification, yet with considerably simpler rule lists. Other variants of the algorithm have been proposed in [16, 17]. Lately data mining using ant-colony optimization has been applied to weed risk assessment [18], credit scoring [19], and prediction of errors in software modules [20]. A common advantage among ant-colony based data mining approaches is their ability to induce accurate, comprehensible and intuitive decision models, which can be augmented by incorporating domain knowledge [20].

## III. Ant Colony Based Spam Filter

In this section, we describe a framework for constructing spam filters based on the Ant-Miner algorithm. The classification rules are extracted from a collection of pre-classified email messages (corpus). Fig. 1 shows the steps involved in constructing the filter model. There are three main procedures: preprocessing and feature extraction, feature selection and training. These procedures are explained next.

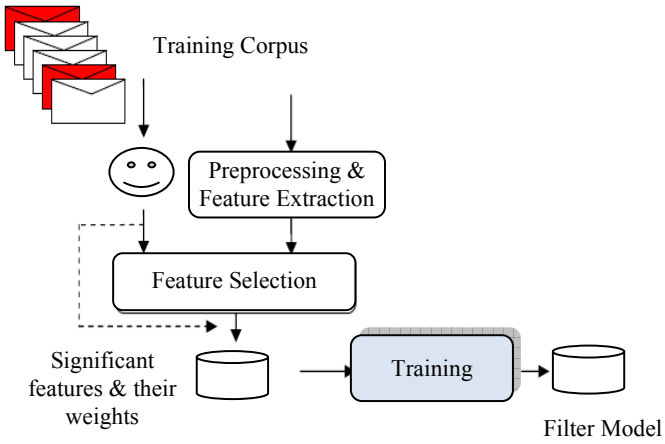


Fig. 1. Model construction for spam filtering.

### A. Preprocessing & Feature Extraction

Each instance message in the corpus is preprocessed to extract representative features. They are determined for each message in the corpus, whether spam or not, by analyzing its content as well as the meta-data associated with it. In order to reduce the high dimensionality and reach optimum results, all HTML tags are first stripped off. Then, all *stop* words, *i.e.* words that appear frequently but have low content discriminating power, are removed from each e-mail message. Examples of such words include 'a', 'an', 'and', 'the', 'that', 'it', 'he', 'she', …, etc. The message is then tokenized into a set of strings separated by some delimiters, *e.g.* whitespaces. These tokens (or terms) can represent words such as 'money', phrases such as 'free gift', or any patterns such as '$$$$'. All mixed-case tokens are converted to lowercase. However, if a token is all uppercase, it will be treated differently than if it is all lowercase. The resulting set of tokens are stemmed to their roots (*i.e.* replacing each token with its base form) to avoid treating different forms of the same word as different attributes; thus reducing the size of the attribute set. For example, both "promotion" and "promoting" are converted to the same base form

"promote". Also if a token appears few times (*e.g*. less than three times) in any email category, it is removed. Capital words in the subject line, fancy headers in the message, and patterns of special characters and punctuations are also good indicators of spam; hence they should be extracted and included in the feature set. The result of this processing phase is that each email message is represented by a feature vector.

*B. Feature Selection*

During this stage, a feature selection technique, such as information gain or chi-square, is applied to the resulting dataset of the previous phase to further reduce the high dimensionality of the feature space. Hence, only candidate features with high weights are kept. The feature vector can be augmented with domain knowledge before and after feature selection. Now, each message is described by a set of attributes ($a_1$, $a_2$, …, $a_m$, $a_{m+1}$) where $m$ is the number of predictive attributes and $a_{m+1}$ is the message class, *i.e*. spam or legitimate. These attributes can be categorical or numeric.

*C. Training*

Before training starts, all numeric attributes are discretized. As a result, each attribute will have a finite domain of values $a_i \in \{v_{i1}, v_{i2}, ..., v_{ik_i}\}$, where $k_i$ is the number of possible values for attribute $a_i$. By having a collection of pre-classified email instances, a rule-based predictive model can be built and deployed to predict the email category of other messages (whether seen or unseen during training). Table I shows an outline of the ant-colony based email miner called AntSFilter. In essence, it is a rule induction meta-heuristic similar to Ant-Miner algorithm. The generic form of each induced rule is as follows:

$$\text{IF } t_1 \wedge t_2 ..... \text{THEN } c$$

where $\wedge$ is the logical AND operator, $t_i$ is a term of the form *attribute* = *value*, and $c$ is the email category predicted by that rule. To avoid invalid rules, each attribute can be used at most once.

To discover a sufficient number of rules that can be used in classification, the problem is first represented as a directed graph consisting of vertices and edges, as shown in Fig. 2, where undirected edges are bidirectional. Each vertex represents a value $v_{ij}$ for $i$=1 to $m$+1 and $j$=1 to $k_i$; a fictitious node is added to represent the start. Each edge represents a path component, which shows the relation between two vertices. To avoid having the same attribute more than once, vertices of each attribute can be connected to vertices of other attributes. Associated with each vertex $v_{ij}$ a variable $\tau_{ij}$ to represent the current level of pheromone on the path component resulting from adding the term $a_i = v_{ij}$ to a partially constructed rule. When the algorithm starts, all pheromone levels are initialized to the same amount, and the

Table I. Pseudo-code of AntSFilter.

```
TrainingSet ←{all training instances in the email corpus};
DiscoveredRuleList ← []; // empty
while(size(TrainingSet) > MaxUncovered){
   InitializePheromone ();
   converge = false;
   t=1; // ant index
   do{
        Ant t incrementally constructs a rule Rt;
        PruneRule(Rt);
        UpdatePheromone();
        converge=TestConvergence();
        t=t+1;
   }while(t < NumAnts ∧ !converge);
   Rbest ← chooseBestRule({all rules constructed by ants});
   DiscoveredRuleList ← DiscoveredRuleList +{Rbest};
   TrainingSet ←
      TrainingSet – {correctly classified instances by Rbest}
}
```

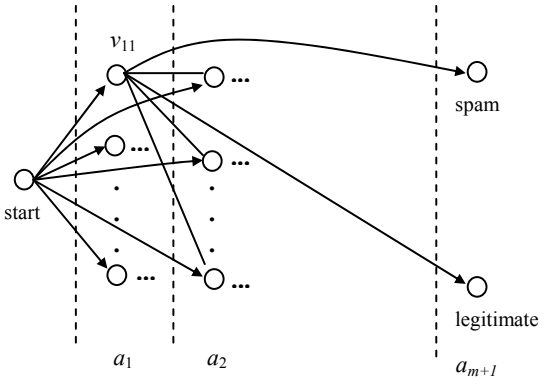


Fig. 2. Partial graph model for ant-colony based spam filter.

training set is initialized to the training instances in the email corpus. The algorithm starts with an empty rule list and iteratively expands it as ants construct rules. In each iteration of the outer loop, the best rule of the rules constructed by various ants is added to the discovered rule list. Each ant starts at the *start* vertex with an empty rule and incrementally constructs a rule by selecting a path to follow. Each time an ant moves to vertex $v_{ij}$, it adds one term, $a_i = v_{ij}$, to its partially discovered rule until all attributes are used or until the number of instances covered by the rule is smaller than a user-specified threshold. Each ant uses a probabilistic selection rule to add the next term to the rule being constructed. The selection probability depends on a problem-specific heuristic $\eta_{ij}$ and the current pheromone level $\tau_{ij}(t)$ associated with the term $a_i = v_{ij}$, as defined by the

following equation:

$$p_{ij} = \frac{\eta_{ij} \cdot \tau_{ij}(t)}{\sum_{i=1}^{m} x_i \sum_{j=1}^{k_i} (\eta_{ij} \cdot \tau_{ij}(t))}$$

where $x_i$ is a binary variable indicating whether the attribute $a_i$ has been already used ($x_i$ = 0) or not ($x_i$ = 1). The problem-specific heuristic, $\eta_{ij}$, is pre-calculated and stored for each term. This heuristic measures the significance or predictive ability of each term. A quantitative measure based on information theory is expressed as follows,

$$\eta_{ij} = \frac{1 - H(C \mid a_i = v_{ij})}{\sum_{i=1}^{m} x_i \sum_{j=1}^{k_i} (1 - H(C \mid a_i = v_{ij}))}$$

where $H(C \mid a_i = v_{ij})$ is the entropy and is calculated from,

$$H(C \mid a_i = v_{ij}) = - \sum_{c \in \{spam, legitimate\}} (P(c \mid a_i = v_{ij}) . \log_2 P(c \mid a_i = v_{ij}))$$

After the rule is completely constructed, it is pruned to remove irrelevant terms. This is done by repeatedly removing a term by term and calculating the rule quality (as defined below). Then the term whose removal results in more improvement in the rule quality is removed. Finally, the pheromone amounts are updated by increasing their levels for all terms in the discovered rule in proportional to the rule quality in predicting the email category of each message in the current training set. The pheromone for all other terms is decreased to mimic evaporation and discourage their selection. The rule quality is computed as the product of the rule sensitivity and specificity, as defined by

$$sensitivity = \frac{TP}{TP + FN}, \quad specificity = \frac{TN}{TN + FP}$$

where *TP* is the number of instances covered by the rule and have the same email category as the category predicted by the rule (*i.e.* true positive), *TN* is the number of instances not covered by the rule and an email category different from the category predicted by the rule (*i.e.* true negative), *FP* is the number of instances covered by the rule but have an email category different from the category predicted by the rule (*i.e.* false positive), and *FN* is the number of instances not covered by the rule and have the same email category as the category predicted by the rule (*i.e.* false negative). The procedure is repeated with a reduced training set formed by removing the instances covered by the best discovered rule, until the number of instances uncovered by the discovered rule list is smaller than a user-specified threshold or until a maximum number of rules have been constructed.

### *C. Spam prediction*

Having a spam filter model constructed as explained in Section III.C, it can now be tested and then deployed to predict the email category for each message. The rules in the discovered rule list are applied in the same order of their discovery until the message is covered. Then, the message will be labeled by the predicted category of the rule that covers the message. If no rule covers the message, then a default rule will be applied (this rule is constructed according to the majority of uncovered instances during training). Fig. 3 shows the model deployment. The user may give feedback to update the filter.

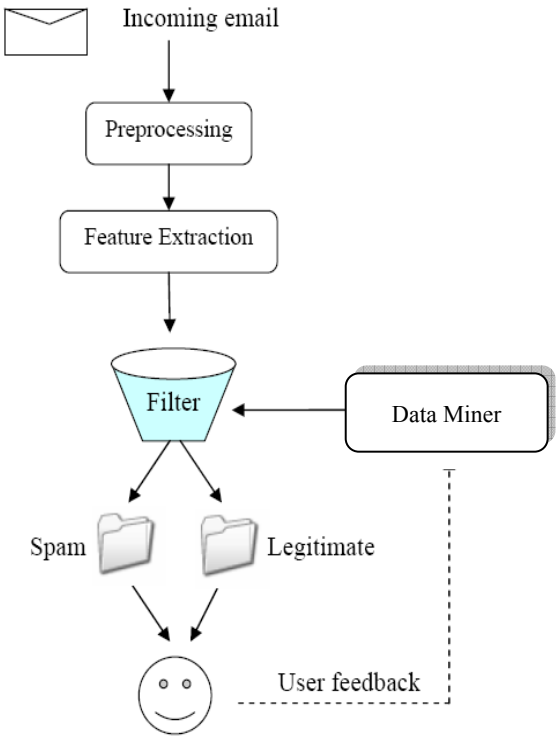


Fig. 3. Model deployment for filtering spam.

## IV. DATA

The AntSFilter is evaluated on a public dataset from the UCI Machine Learning Repository, known as spambase [21]. This dataset has been used in some earlier publications, *e.g.* [8]. The dataset consists of 4601 instances of legitimate and spam email messages with 39.4% being spam. Each instance is characterized by 57 input attributes and is labeled as spam (represented as 1) or legitimate (represented as 0). The attributes include the frequency of various words (*e.g.* "money"), the frequency of special characters (*e.g.* dollar sign), and the length of sequences of consecutive capital letters in the message. Attributes 1-48 give the percentage of words in the email message for the respective keyword indicated in the attribute name. Attributes 49-54 give the percentage of characters in the email message for the respective character indicated in the attribute name. Attributes 55 and 56 give the average and maximum lengths, respectively, of uninterrupted sequences of capital letters in the message. Attributes 57 gives the total number of capital letters in the message. The attribute number will be used as

the variable number for models described throughout this paper. Attribute number 58 in the dataset is the true class (legitimate = 0, spam = 1).

## V. Simulation and Results

To evaluate the effectiveness of the proposed approach, we built a prototype in MATLAB version R2007a and applied it to the spambase dataset (described in Section IV). The dataset is first randomized and preprocessed to convert numeric attributes into discrete ones using two different discretization techniques. The first discretization technique is straightforward where each numeric attribute is split into 10 different equally-spaced intervals between its maximum and minimum values. The second technique uses the minimum description length (MDL) discretization which is a supervised method based on a minimal-entropy heuristic to determine the cutoff points [22]. As a result, about 47.4% of the attributes have two levels (high and low), 28% have three levels (high, medium, low), and the remaining attributes have four to six levels. Table II describes the three datasets used.

Table II. Summary of the datasets.

| Dataset | Num. of instances | Num. of attributes | Description |
|---|---|---|---|
| DS0 | 4601 | 57 | Original dataset with continuous attributes |
| DS1 | 4601 | 57 | Discretized dataset using Equally-spaced intervals |
| DS2 | 4601 | 57 | Discretized dataset using MDL |

We then used 5-fold cross validation to increase the confidence in the results where each dataset is partitioned into five non-overlapping subsets. Then, five experiments were conducted for each dataset. In each experiment, a different subset is used for testing and the remaining four subsets were used for training. At the end, the results were averaged over the five experiments. In all experiments, we had the following five parameter settings for the AntSFilter approach:

- Maximum number of ants for the inner loop = 20
- Minimum number of instances per rule = 5
- Maximum number of uncovered instances = 10
- Number of rules used to test convergence of the ants = 10
- Max number of iterations of the outer loop = 100

Similar approach has been followed in [8] for evaluating MLP and NB on the same dataset. Table III shows the empirical performance measures in terms of the average prediction accuracy. The first column shows the average values for the five experiments conducted in [8]. We can observe that when using AntSFilter with MDL discretization, the prediction accuracy 90.29% is vaguely better than MLP and much better than NB implemented in [8]. But as mentioned, the ant-colony based data miner generates more comprehensive models than neural networks. We also compared the results with another rule-based technique, the Ripper method. Although using Ripper with MDL discretization can give a slightly better prediction accuracy than AntSFilter (see Table III), rules induced by AntSFilter is much simpler as indicated in Table IV.

We have also tested some other parameter settings but without a serious attempt to optimize these settings to make the comparison fair with earlier reported work on the same dataset. For example we tried to increase the number of ants to 100; but the algorithm gets slower since more candidate rules need to be evaluated in each iteration to select the best rule; slight improvement has been observed as well.

Table III. Comparison of predictive accuracy (%) using 5-fold cross-validation

| Method | [8] | DS1 | DS2 |
|---|---|---|---|
| AntSFilter | - | 81.59 | 90.29 |
| MLP | 90.24 | - | - |
| Ripper | - | 79.57 | 91.38 |
| NB | 73.09 | 84.55 | 90.19 |

Table IV. Comparison of rule list size and number of terms per rule using 5-fold cross validation.

| Method | DS1 | | DS2 | |
|---|---|---|---|---|
| | Rules | Terms | Rules | Terms |
| AntSFilter | 6.8 | 9.44 | 7.6 | 3.34 |
| Ripper | 28 | 3.18 | 34 | 4.68 |

## VI. Conclusion and Future Work

In this study, we build an ant-colony based spam filter and evaluate its effectiveness in detecting spam. A framework for the proposed system is described and preliminary results on a publicly available dataset are reported. The results are compared to three machine learning techniques for the same dataset. We also studied the impact of two discretization methods on the performance. We found that the proposed approach can be a promising alternative for spam filtering. Although the discovery of rules takes time (yet better than MLP), the deployment of the discovered rules in detecting spam is very fast which makes it suitable for online deployment. We also found that a large performance improvement can be achieved by controlling the discretization step. We are currently working on further investigation for this approach and evaluating other variants and datasets.

## Acknowledgment

The author would like to thank King Fahd University of

Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia for continuous support and providing computing facilities during this work.

## REFERENCES


[1] B. Hoanca, "How good are our weapons in the spam wars?" *IEEE Technology and Society Magazine*, vol. 25, no. 1, pp. 22-30, 2006.

[2] E.-S. M. El-Alfy, "Learning Methods for Spam Filtering," *International Journal of Computer Research*, vol. 16, no. 4, 2008.

[3] C. C. Wang, "Sender and receiver addresses as cues for anti-spam filtering," *Journal of Research and Practice in Information Technology*, vol. 36, no. 1, pp. 3–7, 2004.

[4] British Computer Society, http://www.bcs.org/.

[5] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[6] G. Sakkis, I. Androutsopoulos, and G. Paliouras, "A memory-based approach to anti-spam filtering," *Information Retrieval*, vol. 6, pp. 49-73, 2003.

[7] J. Provost, "Naïve-Bayes vs. rule-learning in classification of email," The University of Texas at Austin, Department of Computer Sciences, Technical Report AI-TR-99-284, 1999.

[8] Y. Yang, S. Elfayoumy, "Anti-spam filtering using neural networks and Bayesian classifiers," in *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Jacksonville, FL, USA, June 2007.

[9] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in *Proceedings of AAAI'98 Workshop on Learning for Text Categorization*, Madison, WI, July 1998.

[10] E.-S. M. El-Alfy and F. S. Al-Qunaieer, "A fuzzy similarity approach for automated spam filtering," in *Proceedings of IEEE International Conference on Computer Systems and Applications (AICCSA'08)*, Qatar, April 2008.

[11] E.-S. M. El-Alfy and R. E. Abdel-Aal, "Spam filtering with abductive networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'08)*, Hong Kong, June 2008.

[12] B. Leiba and N. Borenstein, "A multifaceted approach to spam reduction," in *Proceedings of First Conference on Email and Anti-spam*, Mountain View, CA, July 2004.

[13] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, 2004.

[14] R.S. Parpinelli, H.S. Lopes and A.A. Freitas, "An ant colony based system for data mining: Applications to medical data," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, USA, July 2001.

[15] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321-332, Aug. 2002.

[16] Q. B. Zhu, and Z. J. Yang, "An ant colony optimization algorithm based on mutation and dynamic pheromone updating," *Journal of Software*, vol. 15, no. 2, pp. 185-192, 2004.

[17] J. Ji, N. Zhang, C. Liu, and N. Zhong, "An ant colony optimization algorithm for learning classification rules," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.

[18] K. Fukuda and J. A. Brown, "Classification rule extraction by Ant-Miner for weed risk assessment," *International Congress on Modelling and Simulation (MODSIM'07)*, Dec. 2007.

[19] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651—665, Oct. 2007.

[20] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M. De Backer, and R. Haesen, "Mining software repositories for comprehensible software fault prediction models," *Journal of Systems and Software*, vol. 81, no. 5, pp. 823-839, May 2008.

[21] UCI Machine Learning Repository, http://mlearn.ics.uci.edu/.

[22] E. J. Clarke, B. A. Barton, "Entropy and MDL discretization of continuous variables for Bayesian belief networks," *International Journal of Intelligent Systems*, vol. 15, pp. 61-92, 2000.