Received 16 December 2014
Revised 7 January 2015
Accepted 11 January 2015

Web spam detection using trust and distrust-based ant colony optimization learning

Bundit Manaskasemsak and Arnon Rungsawang
Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand

Abstract

Purpose – This paper aims to present a machine learning approach for solving the problem of Web spam detection. Based on an adoption of the ant colony optimization (ACO), three algorithms are proposed to construct rule-based classifiers to distinguish between non-spam and spam hosts. Moreover, the paper also proposes an adaptive learning technique to enhance the spam detection performance.

Design/methodology/approach – The *Trust-ACO* algorithm is designed to let an ant start from a non-spam seed, and afterwards, decide to walk through paths in the host graph. Trails (i.e. trust paths) discovered by ants are then interpreted and compiled to non-spam classification rules. Similarly, the *Distrust-ACO* algorithm is designed to generate spam classification ones. The last *Combine-ACO* algorithm aims to accumulate rules given from the former algorithms. Moreover, an adaptive learning technique is introduced to let ants walk with longer (or shorter) steps by rewarding them when they find desirable paths or penalizing them otherwise.

Findings – Experiments are conducted on two publicly available WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets. The results show that the proposed algorithms outperform well-known rule-based classification baselines. Especially, the proposed adaptive learning technique helps improving the *AUC* scores up to 0.899 and 0.784 on the former and the latter datasets, respectively.

Originality/value – To the best of our knowledge, this is the first comprehensive study that adopts the ACO learning approach to solve the problem of Web spam detection. In addition, we have improved the traditional ACO by using the adaptive learning technique.

Keywords Trust, Adaptive learning path, Ant colony optimization, Distrust, Spam detection, Web spam

Paper type Research paper

1. Introduction

Web search engine has become an indispensable tool for finding users' information needs on the Internet. Against billions of Web pages, only the first few search results have a high possibility to be clicked and visited by the users. Those Web pages earn high impacts on commercial, social or political purposes, as a high order in ranking provides large, free advertising as well as increases Web traffic volumes at the same time. In consequence, many Web engineers have put hard efforts to boost a ranking order of their Web pages. However, many Web pages have intentionally been much

The initiative idea of this paper has been previously explored and published in ICCSA2014 conference; the authors thank Mr Apichat Taweesiriwate and Mr Jirayus Jiarpakdee, our former students, for their contribution on the first implementation of the algorithms.

International Journal of Web Information Systems
Vol. 11 No. 2, 2015
pp. 142-161

1744-0084
DOI 10.1108/IJWIS-12-2014-0047

manipulated to get higher ranks which they are not deserved. This kind of deceptive attempt which violates the search engine guidelines has been known as "Web spamming".

Over the past decade, counter-attack research on Web spam has gained a lot of interest both from academia and industry (Castillo *et al.*, 2008). Spam pages will not only degrade quality of search results but also cause the search engines to waste amount of computational and storage resources without benefit. Spamming techniques (Gyöngyi and Garcia-Molina, 2005) can be considered as having three types. First, *content spam* includes techniques that retouch content of target pages, for instance, by inserting a number of keywords that are possibly related to query terms than to their semantic content. Second, *link spam* consists of a creation of link structure to take advantage of link-based algorithms, such as PageRank (Page *et al.*, 1999) and HITS (Kleinberg, 1999) to boost a ranking score of a target page. Last, *hiding spam*, including cloaking and redirection, attempts to deliver different content to normal Web users and search engine Web crawlers.

Several studies on Web spam detection have been proposed for years in different ways, including content-based techniques, e.g. work by Fetterly *et al.* (2004) and Ntoulas *et al.* (2006); link-based ones; work by Wu and Davison (2005), Gyöngyi *et al.* (2004), Wu *et al.* (2006), Becchetti *et al.* (2006), and Castillo *et al.* (2007); and other learning based on browsing logs (Liu *et al.*, 2008a) and user behavior (Liu *et al.*, 2008b). In this work, we concentrate on link-based spam detection scheme and propose an alternative learning approach. By adopting the ant colony optimization (ACO) algorithm (Dorigo *et al.*, 1996, 1999), the model uses a host graph, constructed from a set of Web hosts defined as nodes and their aggregated hyperlinks over Web pages defined as edges, to generate a set of rules from ant trails. Relying on the approximate isolation principle (Gyöngyi *et al.*, 2004) that good (non-spam) pages – hosts in our case – seldom link to bad (spam) ones, it can be implied: the *trust hypothesis* that good pages in general point to good ones and the *distrust hypothesis* that pages pointing to bad one are usually bad themselves. We thus introduce three main approaches that use those hypotheses for Web spam detection: trust- and distrust-based ACO learning and their combination. In the learning phase, we explore the training datasets from both WEBSPAM-UK2006 and -UK2007 collections (Castillo *et al.*, 2006).

For the *Trust-ACO* learning, each Web host labeled with "non-spam" is used as a seed for ants. Trails on the host graph discovered by ants can be referred to trust (non-spam) paths which are subsequently used to generate rules for classifying non-spam hosts from spam ones. Considering each non-spam path, a classification rule is determined by choosing common overlapping characteristic features of all non-spam hosts, and assigned as the "non-spam" class. Similarly, the *Distrust-ACO* learning lets ants start with seeds labeled with spam. Trails discovered by ants are interpreted and then compiled to spam classification rules, used for classifying spam from non-spam hosts. Last, the *Combine-ACO* learning takes the advantages from previous two learning algorithms by accumulating their rules and reordering them. In addition, we propose an adaptive version of the ACO learning that can take a reward or a penalty action to ants to adaptively extend or shorten their walking trails. Experiments conducted on the testing datasets from both standard collections reveal that our approaches can provide more accurate identification of spam and non-spam hosts than well-known rule-based classification baselines.

The remainder of this paper is organized as follows. Section 2 briefly reviews some previous machine learning-based studies. Section 3 introduces the motivation of this work and describes some preliminaries, including a short explanation of the ant colony optimization algorithm. Sections 4 and 5 detail the proposed link-based ACO algorithms and the additional adaptive one. Section 6 reports the analysis of parameter sensitivity and provides performance evaluation. Finally, Section 7 concludes the paper.

2. Related work

Various approaches have been proposed to combat Web spamming and to detect spam pages. Because our ACO approach which simulates the collective intelligent behaviors of ants is one kind of the swarm intelligence techniques, we then mention some previous Web spam detection research that are related to the machine learning techniques here.

Almost machine learning-based methods have considered Web spam detection as a problem of binary classification. Preliminarily, some Web pages, referred to a training dataset, are examined and labeled as spam or non-spam by an expert. Then, a classification model is created by adopting any supervised learning algorithm to learn from these training data. Further, the model is used to predict any new Web pages as either spam or non-spam. The key issue is what features are defined to represent pages used in both learning and classifying processes. For example, we can consider a number of link-based features such as scores derived from PageRank, TrustRank and truncated PageRank computation (Becchetti *et al.*, 2006); the use of content-based features (Ntoulas *et al.*, 2006); and the mixture of both these kinds (Becchetti *et al.*, 2008).

Erdélyi *et al.* (2009) focused on a selection of features from the Internet archives to best describe spam. Araujo and Martinez-Romo (2010) explored different combinations of qualified link-based and language model-based features. They reported to obtain significant improvement in their Web spam detection using fewer numbers of features. An important aspect of this work is that their experimental results were done on the same standard WEBSPAM-UK2006 and -UK2007 datasets as ours. Work by Dong and Zhou (2012) examined several novel topical diversity measures for their content spam classification. Recently, Goh *et al.* (2013) used the multi-layer perceptron neural network and support vector machine (SVM) and evaluated their results on the same standard collections too, while Luckner *et al.* (2014) explored lexical based features trained on WEBSPAM-UK2006, but tested on -UK2007 data during their evaluation.

3. Motivation and preliminaries

The main idea behind our work comes from TrustRank (Gyöngyi *et al.*, 2004), an important and well-known algorithm for combating Web spam. TrustRank was proposed to propagate trust from a small selected seed set of good pages to others via personalized PageRank. The algorithm relies on the *approximate isolation* principle of the good set, that is "good (or non-spam) pages seldom point to bad (or spam) ones", and proceeds as follows. Top-k pages returned from an inverse PageRank computation are first judged by a human expert; pages annotated as good are included in the seed set. Then, a personalized vector is constructed in which all elements corresponding to those good judged pages are assigned to a non-zero value. The trust propagation applied with the personalized vector is afterwards computed recursively from the seed set to the

entire Web with a certain number of iterations. Finally, the algorithm is expected to discover other good pages with higher trust score than that of the bad ones.

Anti-TrustRank (Krishnan and Raj, 2006) is another approach proposed in a manner opposite to TrustRank. The algorithm relies on an implication of the same approximate isolation principle that pages pointing to spam ones are likely to be spam themselves. It thus propagates distrust from a set of known spam pages to the entire Web in the reverse direction of their hyperlinks.

Likewise, our approaches proposed here also rely on the approximate isolation principle. However, instead of assigning a trust/distrust score to each Web page to distinguish the bad from good ones like Trust and Anti-TrustRank, the key difference is to use a machine learning technique, i.e. ACO, to construct a classifier. In our case, we represent the problem as a directed host graph. Given a set of labeled seed set, by adopting ACO on the graph, our learning models make an effort to discover both trust trails when ants move forward from non-spam hosts, and distrust trails when ants move in reverse direction from spam ones. The trails are afterwards interpreted as useful classification rules. In the followings, we preliminarily describe the host graph representation. Then, we briefly provide the concept of ACO before presenting details of the proposed algorithms in the next section:

- We define the host graph as $\mathcal{G}_{\mathcal{H}} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ refer to a set of hosts and their hyperlinks, respectively. A directed edge, $e(h_i, h_j) \in \mathcal{E}$ is defined if there is a Web page u belonging to host $h_i \in \mathcal{V}$ having a link to a page v belonging to the host $h_j \in \mathcal{V}$. However, multiple edges from h_i pointing to h_j will be collapsed into one edge. Self loops are all omitted.
- The ACO algorithm (Dorigo *et al.*, 1996, 1999) is devised to simulate the behavior of ants, including mechanisms of cooperation, to solve the real-world complex problems such as the traveling salesman problem (Dorigo and Gambardella, 1997). By nature, in a colony of social ants, each ant normally has its own duty and performs its own tasks independently from other members of the colony. However, tasks done by different ants are usually related to each other in such a way that the colony, as a whole, is capable of solving complex problems through cooperation. For example, many survival-related problems such as selecting the shortest walking path, finding and storing food, which require sophisticated planning, are solved by an ant colony without any of supervisor. The extensive study from ethologists reveals that ants communicate with others by means of pheromone trails of which path should be followed. As ants move, a certain amount of pheromone is dropped to make the path with the trail of this substance. Ants tend to converge to the shortest trail (or path), as they can make more trips, and hence, deliver more food to their colony. The more ants follow a given trail, the more attractive this trail becomes to be followed by other ants. This process can be described as a positive feedback loop, in which the probability that an ant chooses a path is proportional to the number of ants that has already passed through that path.

Artificial ants of the ACO algorithm solve a problem based on the following concept:

- Each path followed by an ant is associated with a candidate solution for a given problem.

- When an ant follows a path, it drops varying amounts of pheromone on that path in proportion to the quality of the corresponding candidate solution for the target problem.
- Path with a larger amount of pheromone will have a greater probability to be chosen to follow by other ants.

In solving an optimization problem with ACO, we have to determine three following functions appropriately to help the algorithm to get faster and better solutions. The first one is a problem-dependent heuristic function "η" which measures the quality of items (i.e. attribute-value pairs) that can be added to the current partial solution (i.e. rule). The second one is a principle for pheromone updating "τ" which specifies how to modify the pheromone trail. The last one is a probabilistic transition rule "P", based on the value of heuristic function and on the content of pheromone trail, that is used to iteratively construct the solution.

4. ACO learning for Web spam detection

In this section, we present details of our three Web spam detection algorithms, named later *Trust-ACO*, *Distrust-ACO* and *Combine-ACO*, respectively.

4.1 Trust-based ACO learning

The idea of adopting the ACO in the problem of Web spam detection has been first explored by Taweesiriwate *et al.* (2012) and Manaskasemsak *et al.* (2014). However, we here introduce an enhance version, called *Trust-ACO* algorithm, which is based on the trust hypothesis, i.e. an implication that non-spam pages usually link to non-spam ones. The learning model is expected to construct trust classification rules that efficiently distinguish non-spam from spam pages (or hosts in our case).

Given a host graph derived from a training dataset, a host h_i labeled with *non-spam* will be assigned as a seed for an artificial ant to start walking. In each step, the ant that visits at host h_i will randomly choose a link $e(h_i, h_j)$ and afterward move to that target h_j. The edge chosen is dependent on the heuristic value and the pheromone information. Let $P_{ij}(t)$ denote a probability assigned to link $e(h_i, h_j)$ at iteration time t. This probability that guides an ant to randomly walk from a current host h_i to the next host h_j is defined as:

$$P_{ij}(t) = \begin{cases} x_j \cdot \dfrac{\eta_{ij}\tau_{ij}(t)}{\sum_{h_k \in F(h_i)} \eta_{ik}\tau_{ik}(t)} & \text{if } e(h_i, h_j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where η_{ij} and $\tau_{ij}(t)$ are the heuristic value and the pheromone information at iteration time t, respectively. $F(h_i)$ refers to a set of hosts that h_i links to. And, x_j is an indicator used to avoid a cycle of the path; in other words, a host h_j will be visited only once. This indicator is defined as:

$$x_j = \begin{cases} 1 & \text{if } h_j \notin \Gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

If h_j has been chosen to visit, it will be incrementally included in the trail Γ.

The heuristic value η_{ij} defined here can be viewed by two aspects, called later *OutDegree* and *ContentSim* heuristics. First, we hypothesize that the number of multiple links pointing out from a non-spam host h_i can heuristically guide ants to fast discover a next non-spam host h_j. The larger number of out-links the non-spam h_i points to h_j, the higher probability the host h_j will be non-spam. Let e_{ij} be the number of hyperlinks aggregated over all Web pages belonging to the host h_i pointing to any Web pages belonging to the host h_j. Thus, the heuristic value expressed in equation (1) can be calculated by the proportion to the amount of links pointing out:

$$\eta_{ij} = \frac{e_{ij}}{\sum_{h_k \in F(h_i)} e_{ik}}, \tag{3}$$

Second, in a general situation, an author – non-spammer – of a Web page intentionally create a link pointing to a target page which is related to some extent, such as containing similar content or providing further detail. We therefore hypothesize that the similarity of content can heuristically guide the ants to move from a non-spam host to the next non-spam one, as well. In this study, we use a number of content-based features provided by experts (Castillo *et al.*, 2006), instead of an actual content, to explain a host. Let $\boldsymbol{f}_i$ be a feature vector represented the host h_i. Then, the heuristic value expressed in equation (1) can alternatively be calculated by:

$$\eta_{ij} = \frac{Sim(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\sum_{h_k \in F(h_i)} Sim(\boldsymbol{f}_i, \boldsymbol{f}_k)}, \tag{4}$$

Here, *Sim*() can simply be the cosine similarity function used in the classical information retrieval (IR) (Baeza-Yates and Ribeiro-Neto, 1999). Notice that both heuristic functions will assign a constant value to every link; therefore, they can be pre-calculated before starting the ACO learning process.

The other key element of ACO is the pheromone information. Because the ACO algorithm iteratively finds the optimal solution, the pheromone value expressed in equation (1) needs to be changed after each run. The pheromone updating is achieved by the following two fractions: evaporation and reinforcement. The former decreases the pheromone level of each trail by a factor ρ. Typical values for this factor are suggested in the range [0.8, 0.99] in *MAX-MIN* ant system (Stützle and Hoos, 2000). The latter increases the pheromone level by a factor σ only to the best ant's path examined among the entire ones after an iteration. Hence, the pheromone function can be formulated as:

$$\tau_{ij}(t) = \begin{cases} \dfrac{1}{|\mathcal{E}|} & \text{if } t = 1, \\ \rho \cdot \tau_{ij}(t-1) + \sigma \cdot \tau_{ij}(t-1) & \text{otherwise.} \end{cases} \tag{5}$$

For the first iteration, the pheromone information is initially set to the same value over the entire links in the graph. Subsequently, after each iteration run, results of the ACO learning will be expressed by ants' paths that are further interpreted and compiled to classification rules. The reinforcement factor σ, in this study, is determined by the *F-measure* metric that evaluates an obtained classification rule over the training dataset.

Let *CH* be a set of hosts covered by the rule, and *NH* be a set of non-spam hosts. Then, the reinforcement factor is calculated by:

$$precision = \frac{|CH \cap NH|}{|CH|},\ recall = \frac{|CH \cap NH|}{|NH|},\ \sigma = \frac{2 \times precision \times recall}{precision + recall}. \quad (6)$$

Note that we will later provide the detail of the rule construction process, but first describe the pseudo-code of the spam-detection algorithm, illustrated in Figure 1.

As per the algorithm shown in Figure 1, we aim to design a generalized ACO learning framework for our three approaches mentioned here and in the next two subsections. For the *Trust-ACO* algorithm, we have to first select only *non-spam* hosts from the training dataset and include them in the initial seed set for ants to start at line 1 in the main function *general_ACO*. In addition, this main function also requires three pre-defined parameters: the number of iteration runs, ants and maximum hops, respectively. The first parameter, repeating the learning process with the number of iterations at line 3, is needed for guaranteeing that the model will indeed contribute the best classification rule. The second one, defining the number of ants at line 4, directly affects the number of generated rules. In general, the larger the value of this parameter is set, the higher possibility a good path will be found. The last one, an argument passing to the function *walk* at line 5, determines how far an ant can move away from an initial seed. This parameter will affect properties, i.e. either specificity or generality, of a generated rule. Note that we have further provided the sensitivity analysis of these parameters in Subsection 6.2.

More precisely, by invoking the sub-module *walk* at line 5, each ant (i.e. line 15) will be assigned to start walking from a non-spam seed in the *forward* direction of hyperlinks. Considering the host graph $\mathcal{G}_{\mathcal{H}}$, the ant once stepping on the host h_i then, at iteration time

function $general\_ACO(\mathcal{G}_{\mathcal{H}})$
1: **for each** seed host h in $\mathcal{G}_{\mathcal{H}}$ **do**
2: initialize heuristic, pheromones, and probabilities of the whole edges
3: **for each** iteration t **do**
4: create a number of ants
5: let an individual ant invoke the function $walk(nHops)$
6: generate rules from ants' paths and choose the (local) best one
7: adjust the pheromone levels
8: update probabilities of edges
9: delete all ants
10: **end for**
11: consider all local best rules and choose the (global) best one
12: collect the global best rule in the answer set
13: **end for**
14: **return** the sequence of classification rules

function $walk(nHops)$
15: let an ant start from a seed host, temporarily defined as h_i
16: **while** $nHops$ is not equal to 0 **and** ant does not reach the end of path **do**
17: randomly select a next host h_j based on the pre-calculated probabilities $P_{ij}(t)$ and move to that target
18: decrease $nHops$ by 1
19: **end while**
20: **return** the ant's path

Figure 1.
The general *ACO* algorithm for spam detection

t, will randomly choose and follow a link from among h_i's out-links to a next host h_j with the probability $P_{ij}(t)$. This moving step (i.e. line 17) is iteratively proceeded with a certain number of hops or until no more way, excluding a circle, to step on. Eventually, at line 20, the function returns an ant's path that will be used to further construct a *non-spam* classification rule.

We now proceed to the detail of the rule construction process. As it can be seen at line 6, all paths discovered by ants are subsequently interpreted and compiled to useful classification rules. The rule is expressed in a simple form; that is, if *rule antecedent* then *rule consequent*, where the rule antecedent is a conjunction of feature terms and the rule consequent is a predictive class.

In the training dataset, characteristic features and a labeled class of each Web host have been already determined by human experts (Castillo *et al.*, 2006). Given m attribute features $A_1, A_2, \ldots, A_m$ with each feature A_i having n_i possible values $a_{i1}, a_{i2}, \ldots, a_{in_i}$, the predictive rule can be generally expressed by:

$$(A_1 = a_{1j_1}, A_2 = a_{2j_2}, \ldots, A_m = a_{mj_m}) \Rightarrow (Class = \text{either } non\text{-}spam \text{ or } spam),$$

where $j_1, j_2, \ldots, j_m$ are any corresponding indices.

Based on the approximate isolation principle (Gyöngyi *et al.*, 2004), a path, starting from a non-spam host and being discovered along the forward link direction by an ant, should be mainly represented as a sequence of visited non-spam hosts. However, that path may contain some spam, as, in fact, a non-spam page possibly has some hyperlinks to spam ones by spammers' intentional tricks (Gyöngyi and Garcia-Molina, 2005) or the author's unintentional mistake. We therefore consider only the non-spam hosts within the path. To construct a classification rule, the rule antecedent is determined by a list of pairs feature A_i and its value corresponding in common among all those non-spam hosts, while the rule consequent is the certain non-spam class. Let $\Gamma_{p(NH)}$ be a set of all non-spam hosts excerpted from the p-th ant's path. The rule interpreted from that path is constructed by:

$$\text{COMMON}_{\forall h \in \Gamma_{p(NH)}}(A_1 = a^h_{1j_1}, A_2 = a^h_{2j_2}, \ldots, A_m = a^h_{mj_m}) \Rightarrow (Class = non\text{-}spam), \tag{7}$$

where $a^h_{ij_i}$ presents the common value corresponding to the feature A_i given from all non-spam hosts h. Note that if a feature has a value range, then the common value is simply determined by overlapping of those ranges; otherwise, if there is not a value in common, a don't care term "?" will be assigned instead, indicating that feature is not affected in the rule.

Again, the procedure, at line 6 in Figure 1, will then select the best rule from all generated ones by evaluating them over the training data using the *F-measure* metric mentioned in equation (6). This best rule is marked as a local candidate. Afterward, only one best candidate is selected and accumulated into the classification model (i.e. lines 11-12). Eventually, all rules returned at line 14 are sorted in decreasing order of their *F-measure* values, meaning that they are intended to be interpreted in a sequential order during the classification process.

4.2 Distrust-based ACO learning

Relying on the trust hypothesis, the trust-based ACO classifier is constructed to recognize trustworthy pages on the Web. In other words, in the case of Web spam, the model is capable of distinguishing non-spam pages from spam ones.

In the same way, we next introduce the *Distrust-ACO* algorithm which adopts the ACO learning based on the distrust hypothesis. This hypothesis also comes from the implication of the approximate isolation principle; that is, pages pointing to spam ones are likely to be spam themselves. Hence, we expect the learning classifier to be capable of recognizing untrustworthy pages and efficiently distinguishing spam pages (or hosts) from non-spam ones.

Fortunately, the basic concept of the ACO learning can still be proceeded in the similar manner as previously described in Subsection 4.1, and also follow the general framework given in Figure 1. However, there are three main points:

(1) each ant is allowed to walk via inverse-hyperlinks from an initial spam seed;

(2) a classification rule is constructed based on considering common features of all spam hosts along an ant's path; and

(3) a rule will be included in the classifier if it has much ability to classify spam hosts in the training dataset.

We, consequently, summarize the detail of algorithm as follows.

For a training dataset, we suppose that all hosts are first annotated and a host graph $\mathcal{G}_{\mathcal{H}} = (\mathcal{V}, \mathcal{E})$ is derived. As shown in Figure 1, at line 1 of the main function *general_ACO*, a host labeled as *spam* (temporarily defined as h_i) will be assigned as an initial seed for an ant. At line 2, all edges of the graph are needed to be initialized; the probability $P_{ij}(t)$ defined for an edge $e(h_j, h_i)$ is dependent on the product of the heuristic value η_{ij} and the pheromone information $\tau_{ij}(t)$ at any iteration time t as follows:

$$P_{ij}(t) = \begin{cases} x_j \cdot \dfrac{\eta_{ij}\tau_{ij}(t)}{\sum_{h_k \in B(h_i)} \eta_{ik}\tau_{ik}(t)} & \text{if } e(h_j, h_i) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $B(h_i)$ represents a set of hosts that link to h_i. Notice that the probability value will be assigned to only in-links of h_i. The cycle avoidance x_j is still the same function as defined in equation (2).

The heuristic value η_{ij} can also be pre-calculated by two aspects, called later *InDegree* and *ContentSim* heuristics. Because spammers usually create a large amount of hyperlinks on spam pages pointed to each other, especially in case of a link farm (Gyöngyi and Garcia-Molina, 2005), we therefore hypothesize that the proportion to the number of multiple in-links of a spam host h_i from h_j can guide ants to discover h_j. Thus, the former heuristic is defined as:

$$\eta_{ij} = \frac{e_{ij}}{\sum_{h_k \in B(h_i)} e_{ik}}, \tag{9}$$

where e_{ij} means the number of multiple links that h_j points to h_i. For the latter heuristic, although we believe that spam pages linking to each other, in general, are not concerned about their related subject, we still define it here for the performance comparison reason:

$$\eta_{ij} = \frac{Sim(\boldsymbol{f}_i, \boldsymbol{f}_j)}{\sum_{h_k \in B(h_i)} Sim(\boldsymbol{f}_i, \boldsymbol{f}_k)}, \tag{10}$$

where $\boldsymbol{f}_i$ represents a content-based feature vector of the hosts h_i, and $Sim()$ is the cosine similarity function.

For the pheromone information, it is still defined as the same formula shown in equation (5). That is, at the first iteration, the pheromone $\tau_{ij}(t = 1)$ assigned on edge $e(h_j, h_i)$ is set to the same value for the whole edges of the graph. For the subsequent iteration, the pheromone on a (best) path will be updated based on the evaporation and reinforcement factors. The first factor ρ is also suggested in the range [0.8, 0.99], while the second factor σ is dependent on the *F-measure* value of that path examined from the training data, calculated by:

$$precision = \frac{|CH \cap SH|}{|CH|}, \; recall = \frac{|CH \cap SH|}{|SH|}, \; \sigma = \frac{2 \times precision \times recall}{precision + recall}, \tag{11}$$

where CH is a set of hosts covered by the rule, and SH is a set of spam hosts.

Again, at lines 4-5 of the algorithm in Figure 1, when a number of artificial ants have been set, an individual ant has started walking from a spam host. By invoking the function *walk*, at lines 15-19, an ant will repeatedly decide and choose to walk to the next host in the reverse direction of the graph. At line 20, the function finally returns a *distrust* path.

In the process of a rule construction, a path returned by an ant is expected to present a list of spam hosts; however, it may contain some non-spam due to either spammers' intention or authors' mistake. We therefore consider only spam hosts within the path. The rule interpreted from the path is constructed by:

$$\text{COMMON}_{\forall h \in \Gamma_{p(SH)}}(A_1 = a^h_{1j_1}, A_2 = a^h_{2j_2}, \ldots, A_m = a^h_{mj_m}) \Rightarrow (Class = spam), \tag{12}$$

where $\Gamma_{p(SH)}$ refers to a set of all spam hosts excerpted from the p-th ant's path, $a^h_{ij_i}$ presents the common value corresponding to the feature A_i given from all spam hosts h and $j_1, j_2, \ldots, j_m$ are any corresponding indices. If there is not a value in common, a don't care term "?" will be assigned instead.

Similarly, the procedure at line 6 will select the local best rule from all generated ones using the *F-measure* metric defined in equation (11). The global best rule is selected again from all local candidates at lines 11-12, and included into the classification model. Finally, at line 14, all classification rules are returned in decreasing order of their *F-measure* values.

4.3 Combining trust and distrust-based ACO learning

We present the *Combine-ACO* algorithm that combines advantages from both trust- and distrust-based ACO learning. Previously, the *Trust-ACO* algorithm is designed to construct a classifier that efficiently recognizes trustworthy hosts, while *Distrust-ACO* aims to recognize untrustworthy ones. Therefore, we believe that the combination of both algorithms would be complementary to each other.

The *Combine-ACO* algorithm still follows previous procedures as mentioned in Subsections 4.1 and 4.2. In addition, it first accumulates both trust and distrust classification rules, returned at line 14 of Figure 1, and afterward reorders those rules together in decreasing order of their *F-measure* values before further exploiting them in the classification process. We suggest here two variations of the algorithms: *Combine-ACO* with *Degree* and *Combine-ACO* with *ContentSim*. The former refers to the combination of *Trust-ACO* algorithm using *OutDegree* heuristic and *Distrust-ACO* algorithm using *InDegree* one. The latter refers to the combination of *Trust-ACO* algorithm using *ContentSim* heuristic and *Distrust-ACO* algorithm using *InDegree* one.

5. Adaptive ACO learning approach

One of the important factors of the ACO learning is the maximum number of hops. This parameter determines how far ants can possibly move away from their initial seeds; the distance of an ant's path can affect the specificity (or generality) of a classification rule.

Instead of using a fixed maximum number of hops used in the above *Trust-ACO*, *Distrust-ACO* and *Combine-ACO* algorithms, the key idea of our contribution proposed in this section is that this value can be adaptively changed on the fly by either a *reward* or a *penalty* bias. The former means that we will increase an ant's distance for longer step walking when it chooses and steps on the right way; otherwise, we will decrease its walking distance. We then put the acronym ACO^+ to represent the adaptive ACO learning of the proposed algorithms.

Figure 2 illustrates a generalized algorithm of adaptive ACO learning for all *Trust-ACO*$^+$, *Distrust-ACO*$^+$ and *Combine-ACO*$^+$ learning. The main difference to the algorithm shown in Figure 1 is that, at line 5, an ant will invoke the enhanced walk procedure, named *adaptively_walk* (i.e. lines 15-25), instead.

As illustrated in the function *adaptively_walk*, for the *Trust-ACO*$^+$ algorithm, an ant will start walking from a non-spam host h_i, and subsequently, choose from host h_i's out-links to a next host h_j. After moving one step, the ant's remained distance *nHops* will be normally decreased (i.e. line 18). If the next selected host h_j is also non-spam, the ant will get a reward by increasing a walking step of *nHops* (i.e. lines 19-20). Otherwise, if the ant steps on a spam one, it will be penalized with a decay factor of -1 on the *nHops* (i.e. lines 21-22).

This adaptive learning mechanism is expected to produce many proper trust classification rules, as if an ant discovers many spams, the function will return a short ant's path. That is, a path containing few non-spam hosts will possibly generate a too specific rule (i.e. many conjunctive terms of the rule antecedent) which is usually useless and should be abandoned.

Similarly, for the *Distrust-ACO*$^+$ algorithm, an ant will rather start walking from a spam host h_i and then choose a next host h_j by considering all host h_i's in-links. It will get a reward score if that host h_j is spam; otherwise, a penalty score. Last, the *Combine-ACO*$^+$ algorithm takes advantages from both *Trust-ACO*$^+$ and

```
function general_ACO+(G_H)
 1: for each seed host h in G_H do
 2:     initialize heuristic, pheromones, and probabilities of the whole edges
 3:     for each iteration t do
 4:         create a number of ants
 5:         let an individual ant invoke the function adaptively_walk(nHops)
 6:         generate rules from ants' paths and choose the (local) best one
 7:         adjust the pheromone levels
 8:         update probabilities of edges
 9:         delete all ants
10:     end for
11:     consider all local best rules and choose the (global) best one
12:     collect the global best rule in the answer set
13: end for
14: return the sequence of classification rules
function adaptively_walk(nHops)
15: let an ant start from a seed host, temporarily defined as h_i
16: while nHops is not equal to 0 and ant does not reach the end of path do
17:     randomly select a next host h_j based on the pre-calculated probabilities
            P_ij(t) and move to that target
18:     decrease nHops by 1
19:     if h_j has the same class (either non-spam or spam) as the seed host then
20:         increase nHops by 1        /* reward */
21:     else
22:         decrease nHops by 1        /* penalty */
23:     end if
24: end while
25: return the ant's path
```

Figure 2.
The general ACO^+ algorithm for spam detection

Distrust-ACO$^+$ learning processes by combining and reordering their classification rules.

6. Experiments

6.1 Dataset and evaluation metrics

We used two publicly available Web spam collections (Castillo *et al.*, 2006) based on crawled of the .uk Web domain in May 2006 and May 2007. The former, called WEBSPAM-UK2006, includes 77.9 million Web pages (11,400 hosts) and over 3 billion hyperlinks, while the latter, called WEBSPAM-UK2007, includes 105.9 million Web pages (114,529 hosts) and over 3.7 billion hyperlinks. Both collections were examined and annotated by a group of volunteers; a Web hosts was labeled as "spam", "non-spam" or "borderline". However, in our experiments, we restricted the dataset using only hosts labeled with either spam or non-spam. We randomly selected 66 per cent of spam and that of non-spam hosts from the WEBSPAM-UK2006 dataset to create a training dataset and used the latter 34 per cent for the evaluation. For the WEBSPAM-UK2007, it fortunately has already been divided into training and testing datasets. We report their hosts' statistics in Table I. In addition, there is a set of pre-computed features over the hosts of both collections, grouped into 96 content-based, 41 link-based and 138 transformed link-based features. Because all features have been recorded in continuous-range values, we therefore, use the technique proposed by Fayyad and Irani (1993) to discretize their values into multiple intervals.

For the evaluation metrics, we use three standard measures as frequently used in many machine learning studies: true positive rate (*TPR*), false positive rate (*FPR*) and

the area under the receiver operating characteristic (ROC) curve (AUC), compiled from the confusion matrix illustrated in Table II and formulated as in equation (13):

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN}, \tag{13}$$

That is, TPR denotes an ability of identifying spam correctly; FPR denotes a proportion of incorrect identifying non-spam; and AUC is the normalized unit over the previous measures (Fawcett, 2006).

6.2 Parameter sensitivity analysis

In this subsection, we investigate the effect of parameters specified in both ACO and ACO^+ learning algorithms. First, the effect of the number of iterations, we hypothesize that high-quality paths (or rules) should be repeatedly walked through by ants. Second, the effect of the number of ants, we hypothesize that the larger amount of ants would increase more opportunity to discover high-quality rules. Last, the effect of the maximum number of hops (i.e. *nHops*), as this parameter directly affects the generalization of rules, we believe that the quality of rules is dependent on assigning it to the proper value.

We conducted the experiments based on parameter tuning for all variations of the proposed algorithms on both WEBSPAM-UK2006 and -UK2007 collections. Figures 3-5 illustrate the effect of each individual parameter in term of the AUC measure. Note that we here report only the results obtained from *Combine-ACO* and *Combine-ACO*$^+$ algorithms, as the others also provide results in the similar direction.

To study the effect of the number of iterations on Web spam detection performance of the algorithms, we essentially fix the other two parameters. For this, we first heuristically assign the constant values of 100 and 12 to the number of ants and maximum hops, respectively. The reason why we set to these values will be later discussed by the results shown in Figures 4 and 5. As the results shown in Figure 3, the algorithms deliver better performance – higher AUC is better – as the number of

Table I.
Statistics of the datasets used in the experiments

Data collections	No. of spam hosts	No. of non-spam hosts
WEBSPAM-UK2006		
Training dataset	1,202	2,940
Testing dataset	601	1,469
WEBSPAM-UK2007		
Training dataset	222	3,766
Testing dataset	122	1,933

Table II.
The confusion matrix used in case of Web spam

	Actual class	
Predicted outcome	Spam	Non-spam
Spam	*TP*	*FP*
Non-spam	*FN*	*TN*

iterations is increased. The reason is presumed that larger number of iterations can increase more opportunity for ants to repeatedly walk through some high-quality paths. However, the performance becomes approximately saturated after around 40 and 30 iterations for the WEBSPAM-UK2006 and -UK2007, respectively. This phenomenon shows that the algorithms reach an optimal solution, and no much effect for the longer run. Consequently, we will practically assign the number of iterations with a constant value of 40 for all later experiments.

We further conducted the experiments to study the effect caused by the number of ants by setting the number of iterations and maximum hops to 40 and 12, respectively. As the results shown in Figure 4, increasing the number of ants also provides better performance. Moreover, the results report very fast improvement in the early phase (i.e.

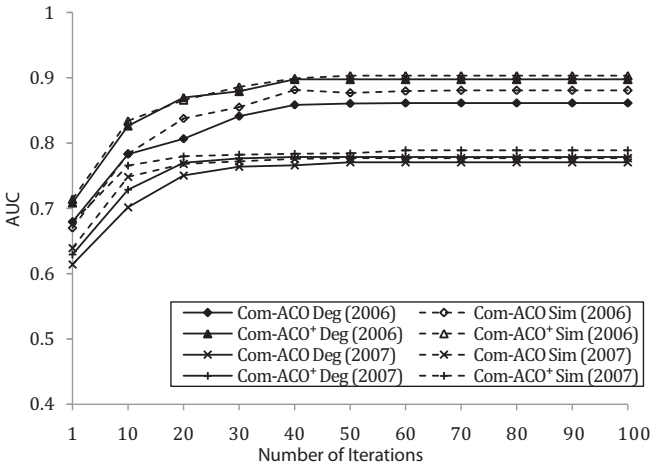

Figure 3. Evolution of *AUC* obtained from applying different number of iterations to *Combine-ACO* and *Combine-ACO*$^+$ algorithms on the WEBSPAM-UK2006 and -UK2007, while the number of ants and maximum hops are constantly set to 100 and 12, respectively

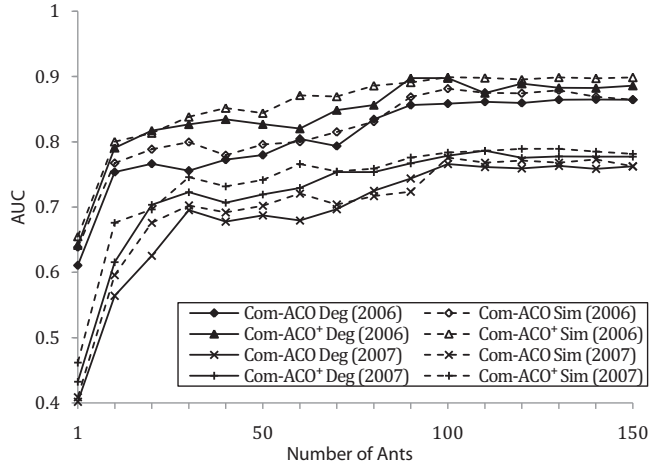

Figure 4. Evolution of *AUC* obtained from applying different number of ants to the *Combine-ACO* and *Combine-ACO*$^+$ algorithms on the WEBSPAM-UK2006 and -UK2007, while the number of iterations and maximum hops are constantly set to 40 and 12, respectively

up to 30 ants), indicating that assigning larger number of ants gives more opportunity to discover several high-quality paths. However, for the later phase, lines in the graph show slightly performance fluctuation after around 100 ants on both datasets. We will then practically set the number of walking ants to 100 for all later experiments.

We lastly study the effect of the maximum number of hops. As the results shown in Figure 5, the algorithms yield very fast improvement as increasing the number of hops in the early phase (i.e. up to 5 hops). The reason is that too short distance – in other words, path – affects to generate a too specific rule (i.e. one that restricts to many features), and thus, may possibly be useless. Therefore, increasing the number of hops for longer walking distance will great relax this restriction. However, the performance of *Combine-ACO* on both datasets becomes decreasing after 12 hops, as the algorithm produces too long paths that even ants select the wrong way. This affects the algorithm to generate too general rules until they could not distinguish between spam and non-spam. In contrast to *Combine-ACO*$^+$, the algorithm can adaptively change the walking distance and also discard the wrong path by a penalty score. In consequence, we will practically set the maximum number of hops to 12 for all later experiments.

6.3 Performance results

We conducted experiments on both WEBSPAM-UK2006 and -UK2007 collections. In the learning phase, we constantly set the number of iterations, ants and maximum hops, used in all variations of *ACO* and *ACO*$^+$ learning algorithms, to 40, 100 and 12, respectively. Moreover, we separated the experiments by training each algorithm using the degree heuristic expressed in equations (3) and (9), and the content similarity heuristic expressed in equations (4) and (10). For the latter heuristic, we only used 96 content-based features to represent a host feature vector. Notice that we, however, still need the entire 275 features for identifying the best rule during the rule construction process.

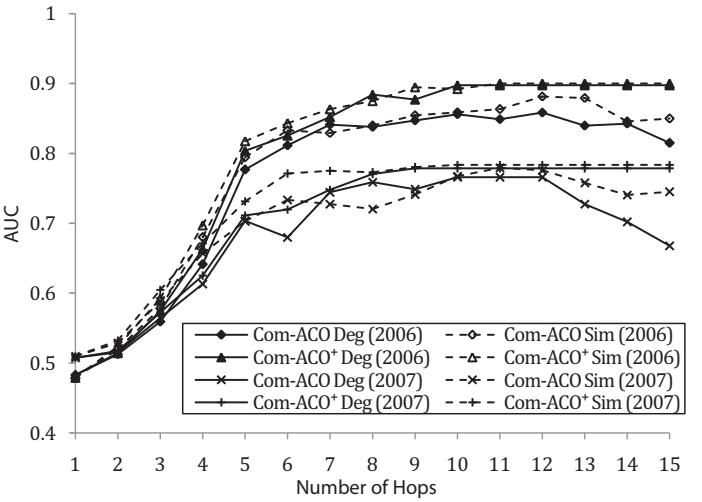

Figure 5.
Evolution of *AUC* obtained from applying different maximum number of hops to *Combine-ACO* and *Combine-ACO*$^+$ algorithms on the WEBSPAM-UK2006 and -UK2007, while the number of iterations and ants are constantly set to 40 and 100, respectively

We compared the performance of our proposed algorithms with four well-known rule-based classification algorithms: decision tree (*C*4.5), *RIPPER*, *PART* and *RandomTree*, respectively, using the same training and testing datasets shown in Table I. We trained those baselines using the WEKA software (Hall *et al.*, 2009). For the environment settings, the rule pruning is enabled; all other parameters have been set to default. The experimental results are concluded in Tables III and IV, based on three standard measures: *TPR*, *FPR* and *AUC*.

As it can be seen in Table III, all variations of *ACO* and *ACO*$^+$ learning algorithms can unanimously deliver better performance on WEBSPAM-UK2006 with much higher *TPR* (the higher, the better), lower *FPR* (the lower, the better) and higher *AUC* (also, the higher, the better) than ones of all four baselines. Furthermore, the *Trust-ACO* and *Trust-ACO*$^+$ algorithms using the *ContentSim* heuristic can produce better performance with higher *AUC* than that of the algorithms using the *OutDegree* heuristic, affirming that linking between trust pages (hosts) has indeed some related content – also, the similar reason for the *Combine-ACO* and *Combine-ACO*$^+$ algorithms. In contrast to *Distrust-ACO* and *Distrust-ACO*$^+$, the algorithms using the *InDegree* heuristic yield better performance than those using the *ContentSim* heuristic due to the fact that spammers, in general, are not interested in content on their spam pages. Lastly,

Algorithm	No. of features	*TPR*	*FPR*	*AUC*
Trust-ACO				
with *OutDegree*	— / 275	0.807	0.189	0.809
with *ContentSim*	96 / 275	0.797	0.134	0.831
Distrust-ACO				
with *InDegree*	— / 275	0.884	0.199	0.842
with *ContentSim*	96 / 275	0.809	0.207	0.801
Combine-ACO				
with *Degree*	— / 275	0.903	0.187	0.858
with *ContentSim*	96 / 275	0.897	0.134	0.881
Trust-ACO$^+$				
with *OutDegree*	— / 275	0.845	0.139	0.853
with *ContentSim*	96 / 275	0.819	**0.087**	0.866
Distrust-ACO$^+$				
with *InDegree*	— / 275	0.900	0.135	0.883
with *ContentSim*	96 / 275	0.889	0.188	0.850
Combine-ACO$^+$				
with *Degree*	— / 275	**0.920**	0.125	0.897
with *ContentSim*	96 / 275	**0.920**	0.122	**0.899**
*C*4.5	275	0.745	0.322	0.712
RIPPER	275	0.707	0.299	0.704
PART	275	0.709	0.304	0.703
RandomTree	275	0.692	0.294	0.699
$C \cup L \cup LM \cup QL$	291	0.89	0.10	0.88

Note: The bold data highlights the best value of results (i.e., best performance)

Table III. Performance comparisons on the WEBSPAM-UK2006 collection

Algorithm	No. of features	*TPR*	*FPR*	*AUC*
Trust-ACO				
with *OutDegree*	— / 275	0.852	0.353	0.750
with *ContentSim*	96 / 275	0.803	0.257	0.773
Distrust-ACO				
with *InDegree*	— / 275	0.869	0.356	0.756
with *ContentSim*	96 / 275	0.820	0.338	0.741
Combine-ACO				
with *Degree*	— / 275	0.869	0.337	0.766
with *ContentSim*	96 / 275	0.852	0.301	0.776
Trust-ACO$^+$				
with *OutDegree*	— / 275	0.861	0.351	0.755
with *ContentSim*	96 / 275	0.811	0.253	0.779
Distrust-ACO$^+$				
with *InDegree*	— / 275	**0.877**	0.340	0.769
with *ContentSim*	96 / 275	0.836	0.333	0.751
Combine-ACO$^+$				
with *Degree*	— / 275	0.869	0.311	0.779
with *ContentSim*	96 / 275	0.869	0.302	**0.784**
*C*4.5	275	0.082	0.009	0.537
RIPPER	275	0.156	**0.007**	0.575
PART	275	0.164	0.041	0.562
RandomTree	275	0.148	0.036	0.557
$C \cup L \cup LM \cup QL$	291	0.50	0.06	0.76

Note: The bold data highlights the best value of results (i.e., best performance)

Table IV. Performance comparisons on the WEBSPAM-UK2007 collection

the comparison between *ACO* and *ACO*$^+$ learning algorithms shows that the latter gives an improvement in all cases, indicating that the addition of the adaptive walking distance on ants can affect the quality of rules.

For comparison with the other work in the literatures, we also excerpt the result from a well-known spam detection model, named $C \cup L \cup LM \cup QL$ (Araujo and Martinez-Romo, 2010), and put it at the last row in Table III. This model is nearly related to ours, in which the authors had combined the content-based (*C*), the transformed link-based (*L*) and their additional new proposed language-model (*LM*) and qualified-link (*QL*) features to construct a decision tree (*C*4.5)-based classifier. It can be seen that both *Combine-ACO* and *Combine-ACO*$^+$ algorithms using *ContentSim* heuristic give higher *AUC* performance, while using original 275 features during the training process.

In the same way, as the results shown in Table IV, all of our approaches still outperform the four baselines with significant higher *TPR* and *AUC* on WEBSPAM-UK2007. However, those baselines provide much lower *FPR*. The reason is that because both training and testing datasets of the WEBSPAM-UK2007, as illustrated in Table I, contain much larger amount of non-spam hosts than spam ones, the baselines then have recognized most characteristic hosts from the majority

class during the training process. Consequently, they possibly predict most unknown hosts as non-spam, while only a few ones as spam. There are thus only few wrong predictions on the testing data. However, it can be argued that our approaches are still capable of dealing quite well with the imbalanced data, as in the case of the WEBSPAM-UK2007 dataset.

When examining the effect of the heuristics used in our approach, we can see that the *ContentSim* one still gives better performance in all *Trust-* (also, *Combine-*) based *ACO* and ACO^+ algorithms. On the other hand, the *InDegree* heuristic provides better performance in all *Distrust*-based *ACO* and ACO^+ algorithms. Note that the ACO^+ learning algorithm yields a slight improvement over the *ACO* one in all cases.

When comparing performance of our approaches with $C \cup L \cup LM \cup QL$, the *Combine-ACO* and *Combine-*ACO^+ algorithms using the *ContentSim* heuristic can give slightly higher *AUC* scores.

7. Conclusion

In this paper, we investigate the problem of Web spam detection and propose a new methodology that adopts the ant colony optimization learning to construct a rule-based classifier. We have presented three main approaches: *Trust-ACO*, *Distrust-ACO* and *Combine-ACO*, that rely on the trust and distrust hypotheses (i.e. implications of the approximate isolation principle, introduced in TrustRank). The first approach is designed to construct a non-spam classifier, interpreted from ants' trust paths, to distinguish non-spam from spam Web pages, while the second one constructs a spam classifier for distinguishing spam from non-spam pages. The last one takes the advantages of both classifiers by combining together the entire rules and then reordering them. Moreover, we have also proposed an adaptive learning technique (i.e. ACO^+) by giving either a reward or a penalty score to ants in the walking process to enhance the performance.

We evaluate our approaches using two publicly available WEBSPAM-UK2006 and WEBSPAM-UK2007 datasets. Base on studying the parameter sensitivity, the result reveals that increasing the number of iterations and ants can improve spam classification performance, while the maximum number of hops needs to be assigned to a proper value. When comparing with several well-known baselines, the results show that all variations of our proposed approaches give better performance with significant higher *AUC* score. In addition, it has been proven that the utilization of content-based features and the employment of adaptive learning paths for ACO can also improve the performance.

References

Araujo, L. and Martinez-Romo, J. (2010), "Web spam detection: new classification features based on qualified link analysis and language models", *IEEE Transactions on Information Forensics and Security*, Vol. 5 No. 3, pp. 581-590.

Baeza-Yates, R.A. and Ribeiro-Neto, B.A. (1999), *Modern Information Retrieval*, Addison Wesley.

Becchetti, L., Castillo, C., Donato, D., Leonardi, S. and Baeza-Yates, R.A. (2006), "Link-based characterization and detection of web spam", *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web, Seattle, WA*, pp. 1-8.

Becchetti, L., Castillo, C., Donato, D., Leonardi, S. and Baeza-Yates, R.A. (2008), "Web spam detection: link-based and content-based techniques", *The European Integrated Project*

Dynamically Evolving, Large Scale Information System (DELIS): Proceedings of the Final Workshop, Vol. 222, pp. 99-113.

Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M. and Vigna, S. (2006), "A reference collection for web spam", *ACM SIGIR Forum*, Vol. 40 No. 2, pp. 11-24.

Castillo, C., Donato, D., Gionis, A., Murdock, V. and Silvestri, F. (2007), "Know your neighbors: web spam detection using the web topology", *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY*, pp. 423-430.

Castillo, C., Chellapilla, K. and Davison, B.D. (2008), "Adversarial information retrieval on the web (AIRWEB 2007)", *ACM SIGIR Forum*, Vol. 42 No. 1, pp. 68-72.

Dong, C. and Zhou, B. (2012), "Effectively detecting content spam on the web using topical diversity measures", *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, Vol. 1, *Washington, DC*, pp. 266-273.

Dorigo, M., Di Caro, G. and Gambardella, L.M. (1999), "Ant algorithms for discrete optimization", *Artificial Life*, Vol. 5 No. 2, pp. 137-172.

Dorigo, M. and Gambardella, L.M. (1997), "Ant colony system: a cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, Vol. 1 No. 1, pp. 53-66.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996), "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on System, Man, and Cybernetics*, Vol. 26 No. 1, pp. 29-41.

Erdélyi, M., Benczúr, A.A., Masanés, J. and Siklósi, D. (2009), "Web spam filtering in internet archives", *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, *New York, NY*, pp. 17-20.

Fawcett, T. (2006), "An introduction to roc analysis", *Pattern Recognition Letters*, Vol. 27 No. 8, pp. 861-874.

Fayyad, U.M. and Irani, K.B. (1993), "Multi-interval discretization of continuous-valued attributes for classification learning", *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, *Chambéry*, pp. 1022-1027.

Fetterly, D., Manasse, M. and Najork, M. (2004), "Spam, damn spam, and statistics: using statistical analysis to locate spam web pages", *Proceedings of the 17th International Workshop on the Web and Databases*, *Paris*, pp. 1-6.

Goh, K.L., Singh, A.K. and Lim, K.H. (2013), "Multilayer perceptrons neural network based web spam detection application", *Proceedings of the IEEE China Summit & International Conference on Signal and Information Processing*, *Beijing*, pp. 636-640.

Gyöngyi, Z. and Garcia-Molina, H. (2005), "Web spam taxonomy", *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*, *Chiba*, pp. 39-47.

Gyöngyi, Z., Garcia-Molina, H. and Pedersen, J. (2004), "Combating web spam with trustrank", *Proceedings of the 13th International Conference on Very Large Data Bases*, *Stanford, CA*, pp. 576-587.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009), "The weka data mining software: an update", *ACM SIGKDD Explorations Newsletter*, Vol. 11 No. 1, pp. 10-18.

Kleinberg, J.M. (1999), "Authoritative sources in a hyperlinked environment", *Journal of the ACM*, Vol. 46 No. 5, pp. 604-632.

Krishnan, V. and Raj, R. (2006), "Web spam detection with anti-trust rank", *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web*, *Seattle, WA*, pp. 37-40.

Liu, Y., Gao, B., Liu, T.Y., Zhan, Y., Ma, Z., He, S. and Li, H. (2008a), "Browserank: letting web users vote for page importance", *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, *New York, NY*, pp. 451-458.

Liu, Y., Zhang, M., Ma, S. and Ru, L. (2008b), "User behavior oriented web spam detection", *Proceedings of the 17th International Conference on World Wide Web*, *New York, NY*, pp. 1039-1040.

Luckner, M., Gad, M. and Sobkowiak, P. (2014), "Stable web spam detection using features based on lexical items", *Computers & Security*, Vol. 46, pp. 79-93.

Manaskasemsak, B., Jiarpakdee, J. and Rungsawang, A. (2014), "Adaptive learning ant colony optimization for web spam detection", *Proceedings of the 14th International Conference on Computational Science and Its Applications*, *Guimarães*, *30 June-3 July*, pp. 642-653.

Ntoulas, A., Najork, M., Manasse, M. and Fetterly, D. (2006), "Detecting spam web pages through content analysis", *Proceedings of the 15th International Conference on World Wide Web*, *New York, NY*, pp. 83-92.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1999), "The Pagerank citation ranking: bringing order to the web", Technical report, Stanford Digital Libraries.

Stützle, T. and Hoos, H.H. (2000), "Max-min ant system", *Future Generation Computer Systems*, Vol. 16 No. 9, pp. 889-914.

Taweesiriwate, A., Manaskasemsak, B. and Rungsawang, A. (2012), "Web spam detection using link-based ant colony optimization", *Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications*, *Fukuoka*, pp. 868-873.

Wu, B. and Davison, B.D. (2005), "Identifying link farm spam pages", *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, New York, NY, pp. 820-829.

Wu, B., Goel, V. and Davison, B.D. (2006), "Propagating trust and distrust to demote web spam", *Proceedings of the Workshop on Models of Trust for the Web*, *Edinburgh*.

Corresponding author
Bundit Manaskasemsak can be contacted at: un@mikelab.net