2012 26th IEEE International Conference on Advanced Information Networking and Applications

Web Spam Detection using Link-based Ant Colony Optimization

Apichat Taweesiriwate, Bundit Manaskasemsak and Arnon Rungsawang
Massive Information & Knowledge Engineering Laboratory
Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Bangkok 10900, Thailand.
Email: {ball, un, arnon}@mikelab.net

***Abstract*—Web spam is one of the most important problems which degrade quality and efficiency of web search engines. In this paper, we present a novel link-based ant colony optimization learning algorithm for spam host detection. The host graph is first constructed by aggregating pages' hyperlink structure. Following the TrustRank assumption, ants start walking from a normal host and randomly follow host links with a probability distribution. Then, the classification rules are appropriately generated according to common features of normal hosts sequentially discovered by ants. From the experiments with the WEBSPAM-UK2006 dataset, the proposed learning model provides much accuracy in classifying both normal and spam hosts than several baselines, including a state of the art C4.5. Moreover, we also provide an analysis in parameter tuning for better results.**

Keywords - link spam; content spam; web spam; spam detection; ant colony optimization algorithm; web link structure

I. INTRODUCTION

World Wide Web is the most enormous media for sharing information and resources. Millions of users nowadays are processing various transactions on web pages, especially searching information. Web search engine therefore has become an important tool. For a given query, the search engine usually has its own ranking mechanism to provide a list of relevant results to users. Against large amount of web pages on the web, only first ten or twenty resulting pages have high possibility to be examined by the users. When the users see such relevant pages, they will then click and follow them.

For many commercial web sites, currently competitive business gives birth to aggressive attempts from web engineers to boost the ranking of their web pages in order to increase the return of investment (ROI). A raising order in search results can persuade visitors to be interested in. In other word, it may increase sales, incomes as well as profits.

There are several operators in Search Engine Optimization (SEO) techniques that attempt to influence positions in result ranking. By ethical manipulation, called white-hat SEO, these operators try to improve search quality and serve more valuable content to users. On the contrary, the grey-hat or black-hat SEO optimizes the sites using backdoors, cloaking, or other tricks that violate the search engine guidelines. These sites are called "web spam" and those manipulations will degrade the quality of searching. Besides, a large number of spam pages explicitly increase the cost of crawling, and inflate both index and storage with worthlessness.

Gyöngyi and Garcia-Molina described in [12] the varieties of spamming techniques. These include the creation of page content that engineers a number of popular keywords, called "content spam." The effort expects that the target page will have high possibility to be returned and also ranked high position within the search results, based on term relevancy to users' queries. In general, those keywords are quite unrelated to real context of the target page, but usually draw the users to visit. The other technique is the creation of large number of extrinsic web pages that point to a target page, called "link spam." Since the traditional link-based ranking algorithms, such as PageRank [20] and HITS [16], calculate the popularity score—page with large score will be ranked high position—of web pages based on link structure analysis, the effort exploits the weakness of those algorithms by forming a farm of linkage in order to unreasonably increase the score of the target.

Many researchers have concentrated on combating spam. For example, Gyöngyi et al. [13] propose an idea to propagate trust from good sites to demote spam, while as Krishnan and Raj [17] reversely propagate anti-trust from spam sites. Wu and Davison [23] expand from a seed set of spam pages to the neighbors to find more suspicious pages in the web graph. Dai et al. [5] exploit the historical content information of web pages to improve spam classification, while Chung et al. [4] propose to use time series to study the link farm evolution. Martinez-Romo and Araujo [18] apply a language model approach to improve web spam identification.

In this paper, we propose to apply the ant colony optimization (ACO) algorithm [7, 9] to detect spam in host level. The host link structure is first constructed by aggregating hyperlinks over web pages. For the training dataset, each host labeled as "normal" is used as a seed of the ACO algorithm. Based on the assumption claimed in TrustRank [13] that good (normal) pages—hosts in our case—seldom link to bad (spam) ones, trails on the host graph discovered by ants can be referred to good (normal) paths. These paths are subsequently used to generate rules in order to classify normal hosts from spam ones in the testing dataset. Considering each normal path, a classification rule is determined by both common content and link features of among hosts contained in that path. From the experiments with the WEBSPAM-UK2006 [2], the results show that rules generated from ACO learning model can classify spam hosts more precise than the baseline decision tree (C4.5 algorithm).

1550-445X/12 $26.00 © 2012 IEEE
DOI 10.1109/AINA.2012.118

The rest of this paper is organized as follows. Section 2 addresses some related work on web spam detection using machine learning techniques and then overviews the basic concept of the ACO algorithm. Section 3 introduces host graph representation and describes the proposed link-based ACO as well as how to generate classification rules. Section 4 reports performance evaluation and parameter sensitivity analysis. Section 5 finally concludes the paper.

II. Related Work and Background

A. Web Spam Detection using Machine Learning Techniques

Web spam detection became a known topic to academic discourse since the Davison's paper on using machine learning techniques to identify link spam [6], and was further reasserted by Henzinger et al. [14] as one of the most challenges to commercial search engines. Web spam detection can be seen as a binary classification problem; a page or host will be predicted as spam or not spam.

Fetterly et al. [11] observed the distribution of statistical properties of web pages and found that they can be used to identify spam. In addition to content properties of the web pages or hosts, link data is also very helpful. Becchetti et al. [1] exploit the link features, e.g., the number of in- and out-degree, PageRank [20], and TrustRank [13], to build a spam classifier. Following the work in [1, 19], Castillo et al. [3] extract link features from the web graph and host graph, and content features from individual pages, and use the simple decision tree C4.5 to build the classifier. Recently, Dai et al. [5] extract temporal features from the Internet Archive's Wayback Machine [15] and use them to train a cascade classifier built from several SVM^{light} and a logistic regression implemented in WEKA [22].

B. Basic Concept of Ant Colony Optimization

Naturally, distinct kind of creatures behaves differently in their everyday life. In a colony of social ants, each ant usually 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 [9, 21]. For example, for survival-related problems such as selecting the shortest walking path, finding and storing food, which require sophisticated planning, are solved by ant colony without any kind of supervisor. The extensive study from ethologists reveals that ants communicate with one another by means of pheromone trails to exchange information about 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), since 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 [7, 9].

Researchers try to simulate the natural behavior of ants, including mechanisms of cooperation, and devise ant colony optimization (ACO) algorithms based on such an idea to solve the real world complex problems, such as the travelling salesman problem [8], data mining [21]. ACO algorithms 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 amount of pheromone on that path in proportion with 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 choose three following functions appropriately to help the algorithm to get faster and better solution. 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 rule 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 the heuristic function and on the contents of the pheromone trail that is used to iteratively construct the solution.

III. Link-based Ant Colony Optimization for Detecting Spam

The proposed link-based spam detection using ant colony optimization (LSD-ACO) approach does follow the same assumption claimed in [13] that good pages/hosts seldom link to bad ones. This assumption implies that normal hosts usually link to other normal ones, as well. We therefore apply the ACO algorithm on the host graph. Our effort is to discover good trails (i.e., path containing normal hosts) in order to generate useful classification rules.

A. Modeling the Host Graph

In a learning mechanism based on the ACO algorithm, most problems are often modeled as a graph. Therefore, we let $G_H = (V, E)$ be a host graph, where V and E represent a set of hosts and hyperlinks between the hosts, respectively. A host link $e(h_i, h_j) \in E$ if there is a web page u belonging to host $h_i \in V$ having a link to a target page v belonging to host $h_j \in V$ and $i \neq j$. Since the host graph is a higher view aggregated from the web graph, it may contain multiple links between the hosts. Hence, we let $|e(h_i, h_j)|$ be a number of those links out-going from host h_i to h_j. It obviously can be seen that $|e(h_i, h_j)| = 0$ in case of self-linking, i.e., $i = j$.

B. Learning Approach with the ACO Algorithm

Given a host graph, we first assign an artificial ant to start walking from a normal host h_i. Behavior of that ant will decide to choose a path $e(h_i, h_j)$ to walk in each step, from h_i to h_j, using a randomization with the probability distribution based on heuristic and pheromone information values. We let P_{ij} denote a probability assigned to the link $e(h_i, h_j)$. The

probabilistic transition that guides an ant to randomly walk from a current host h_i to any next host h_j is defined as:

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

where η_{ij} and $\tau_{ij}(t)$ denote a heuristic function and a pheromone information function obtained at the iteration time t, respectively, determined for the link $e(h_i, h_j)$. $F(h_i)$ denotes a set of hosts that h_i links to. Last, x_j is an indicator used to avoid repeated hosts h_j (i.e., cycle) within the ant's trail. Let Γ be the set of hosts sequentially discovered by the ant. This indicator is defined as:

$$x_j = \begin{cases} 1 & \text{If } h_j \notin \Gamma, \\ 0 & \text{Otherwise.} \end{cases} \quad (2)$$

Based on our assumption, we believe that the number of multiple links pointed out from a normal host h_i can be heuristically guided ants to find a next normal host h_j. The larger number of h_i's links points to h_j, the higher possibility the h_j is the normal host. We therefore define the heuristic function illustrated in (1) as the proportion to the amount of links pointed out:

$$\eta_{ij} = \frac{|e(h_i, h_j)|}{\sum_{h_k \in F(h_i)} |e(h_i, h_k)|}. \quad (3)$$

Note that the heuristic value assigned to every link is simply constant; therefore, we can pre-calculate before starting the ACO computation.

Since the ACO algorithm iteratively finds the optimal solution, the pheromone in (1) which controls the movement of ants will be changed for each run. The pheromone information function, defined at the iteration time t = 1, 2, 3, …, is calculated by:

$$\tau_{ij}(t) = \tau_{ij}(t-1) + \tau_{ij}(t-1)q, \quad (4)$$

where q measures the quality of prediction rules over the training dataset. This measure is defined as the confidence value. Let CH be a set of hosts covered by the rule, and NH be a set of normal hosts. Then, the quality metric is defined as:

$$q = \frac{|CH \cap NH|}{|CH|}, \quad (5)$$

i.e., the number of hosts covered and correctly predicted by the rule divides by the number of all hosts covered by that rule.

For the first iteration, the initial pheromone values of all links in the graph are equally set to:

$$\tau_{ij}(t=0) = \frac{1}{|E|}. \quad (6)$$

After each iteration run, a result of the ACO learning can be expressed by paths of ant walking. These trails are further needed to be interpreted and generated classification rules. Note that we will give detail of this process in the next subsection. However, we here illustrate the pseudo-code of our LSD-ACO in Algorithm 1.

ALGORITHM 1. THE LSD-ACO ALGORITHM.

```
function LSD-ACO (seeds,    // set of normal hosts
                  nIters,   // number of iterations
                  nAnts,    // number of ants
                  nHops)    // number of hops
 1: rules = ϕ
 2: repeat
 3:    s ← dequeue (seeds)
 4:    τ ← initializePheromone ( )
 5:    bestRules = ϕ
 6:    maxQ = 0
 7:    for t = 0 to nIters – 1 do
 8:       paths = ϕ
 9:       for a = 0 to nAnts – 1 do
10:          path ← walk (s, τ, nHops)
11:          paths ← paths ∪ path
12:       end for
13:       allRules ← generateRuleOnPath (paths)
14:       (curRule, curPath, q) ← selectHighQRule (allRules)
15:       τ ← updatePheromone (q, curPaths)
16:       if maxQ < q then
17:          bestRules = curRules
18:          maxQ = q
19:       end if
20:    end for
21:    rules ← rules ∪ bestRules
22: until seeds is empty
23: return rules

function walk (s,           // starting normal host
               τ,           // pheromone information
               nHops)       // number of hops
 1: path = ϕ
 2: for i = 0 to nHops – 1 do
 3:    r ← randomlySelectTargetHost (s, τ)
 4:    path ← appendLink (path, e(s, r))
 5:    s = r
 6: end for
 7: return path
```

As shown in the algorithm, the function *LSD-ACO* needs four input parameters: a set of starting normal hosts, the maximum number of iteration runs, ants, and hops, respectively. For the outer loop between line 2 and 22, each normal host s is selected and specified as a starting host for ants. In line 4, all pheromone information values are first initialized by (6). The middle loop, between line 7 and 20, iterates the computation until reaching the number of pre-defined rounds. For the inner loop between line 9 and 12, each ant starts walking from s through the host graph. The function *walk* in line 10 returns a trail of an ant, and that will be gathered by line 11. Subsequently, in line 13–14, all discovered paths are interpreted and classification rules are generated; but, only the rules having the highest quality will be selected. Next, the pheromone information values are updated by the qualities q along only those paths producing the selected (quality) rules. The procedure, in line 16–19 and 21, brings the best rules over all iterations as the results.

More precisely for the function *walk*, it determines behavior of ants' walking. Since a host graph may be very large, if the ants are allowed continually walking until reaching the end of path, they will not only spend a lot of time but also have higher chance to discover spams. Therefore, the parameter *nHops* is used to determine the termination of ants' walking. By the loop between line 2 and 6, the target (normal) host r is randomly selected with some probabilities based on the heuristic function and pheromone information described in (1), and that sub-path is appended to obtain the final result.

C. Determining the Classification Rules

As mentioned to the procedures at line 13–14 of the function *LSD-ACO* in Algorithm 1, all paths discovered by ants are needed to be interpreted as well as generated useful classification rules. We first assume that some characteristic features have already been extracted from each host in order to determine its class C, i.e., either normal or spam. Let $\{A_1, A_2, \ldots, A_m\}$ be a set of m host features, and $\{a_{i1}, a_{i2}, \ldots, a_{in_i}\}$ be a set of n_i possible values belonging to the feature A_i. Thus, the association of features and class of a host can be represented by:

$$(A_1 = a_{1x}, A_2 = a_{2y}, \ldots, A_m = a_{mz}) \Rightarrow (C = normal \text{ or } spam),$$

where x, y, and z are any indices.

Following the TrustRank idea [13], a path discovered by an ant should mainly be a list of normal hosts. However, that path may contain some spam ones since in fact a normal host may possibly or unintentionally link to a spam. Therefore, only normal hosts within the path will be considered. To generate a rule, the value for a host feature A_i is determined by common value of that feature of among all those normal hosts. Let Γ_{NH_p} be a set of normal hosts within the path p. The rule corresponding to p is defined as:

$$\underset{h \in \Gamma_{NH_p}}{COMMON}(A_1 = a_{1x}^h, \ldots, A_m = a_{mz}^h) \Rightarrow (C = normal). \quad (7)$$

Note that if a feature has an interval of values, the common value will then be given by an overlap of those intervals. Otherwise, if there is not any value in common, a don't care term "*?*" will be assigned instead, meaning that the feature is not affected in the rule.

After generating all possible rules, the consecutive procedure at line 14 of the function *LSD-ACO* in Algorithm 1 will measure the quality of each rule within the training dataset by using (5). Then, the rule with the highest quality is selected as a candidate.

IV. Experiments and Results

A. Dataset Preparation

We used the WEBSPAM-UK2006 [2] containing 1,803 hosts labeled as spam and 4,409 hosts labeled as normal; all hosts are within the .uk domain. The dataset contains several features, including both content- and link-based features, as well as a spamicity value of each host. We further processed the dataset as follows (see Fig. 1):

- For the 1,803 spam hosts, we first sort them by ascending order of the spamicity values. Each host will be assigned with an identification number beginning from 0. We then decompose these spam hosts into 3 buckets by considering the remainder from dividing its identification number with 3. Eventually, we will have "bucket1", "bucket2", and "bucket3", in which each contains equally 601 spam hosts.
- Similarly, for the 4,409 normal hosts, we sort them by descending order of the spamicity values. We equally divide them into 10 portions, and assign an identification number beginning from 0 to each host in each portion separately. For each portion, the host whose identification number divided by 7 leaves a remainder of 0, 2, and 5, will then be assigned into "bucket1", "bucket2", and "bucket3", respectively. Note that the host with prior order of identification number—from every portion—will be first assigned. To avoid data imbalance of normal and spam hosts in training set, we stop this assigning process if each bucket contains 601 normal hosts. For the whole remaining hosts, we put them into a new "bucket4".

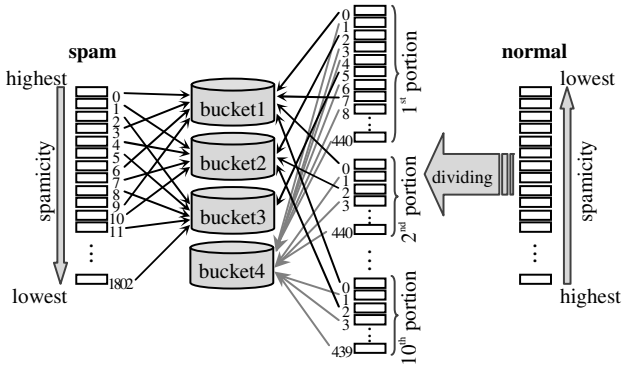

Figure 1. Dataset preparation.

In the LSD-ACO learning model, we used almost all the features, except hostID and hostname, to generate rules. However, since all features have continuous-range values, we therefore used the technique proposed by Fayyad and Irani [10] to first discretize them into multiple intervals.

B. Experimental Results

Using the dataset described in the previous subsection, we designed experimental studies as the following scenarios:

- scenario1: the bucket1 is used for training, while the bucket2, bucket3, and bucket4 are for testing.
- scenario2: the bucket2 is used for training, while the bucket1, bucket3, and bucket4 are for testing.
- scenario3: the bucket3 is used for training, while the bucket1, bucket2, and bucket4 are for testing.

In each scenario, all 601 normal hosts of the training set are used as seeds for ants to train the LSD-ACO model. For each seed, we reiterate the learning process 40 times. Each run is based on 100 artificial ants that walk away from a seed up to 12 hops.

We compared performance of the LSD-ACO algorithm with four well-known rule-based learning models: the decision tree (C4.5), RIPPER, PART, and RandomTree, respectively, using an accuracy measurement, i.e., the correctness of classifying both normal and spam hosts in the testing set. We trained these four baselines using the WEKA software [22]. For the parameter settings, the rule pruning is enabled; however, all remaining ones have been assigned by default. We illustrate the performance comparisons in Fig. 2.

As we can see in the figure, the proposed LSD-ACO learning model outperforms the four baselines. It has the ability to classify both normal and spam hosts with the most accuracy in all experiments.

C. Parameters Sensitivity Analysis

In this subsection, we investigated the effect of parameters specified in the LSD-ACO algorithm. First, the effect of number of iterations (i.e., *nIters*) on the efficiency of rules: we expect that high quality paths should be repeatedly walked through by ants. Second, the effect of number of ants (i.e., *nAnts*) on the number of possible rules: we expect that assigning large amount of ants should increase an opportunity to discover high quality rules. Last, the effect of number of hops (i.e., *nHops*) on the generalization of rules: since long path may possibly lead to many non-overlapping feature values.

We conducted the experiments based on the previous three scenarios. In each scenario, we run the LSD-ACO algorithm at least 8 times and averaged the results. Fig. 3, 4, and 5 as follows illustrate the average accuracies over all scenarios by tuning parameters: the number of iterations, the number of ants, and the number of hops, respectively. In the figures, a point on the line is an average value. We also plot both the worst (i.e., the lowest) and the best (i.e., the highest) values possibly obtained from the experiments.

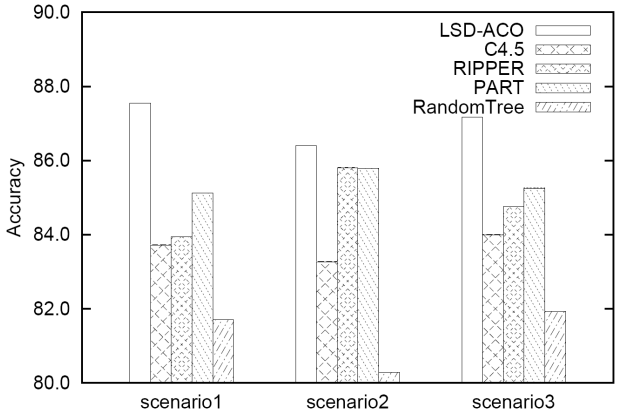

Figure 2. Performance comparisons in classifying normal and spam hosts.

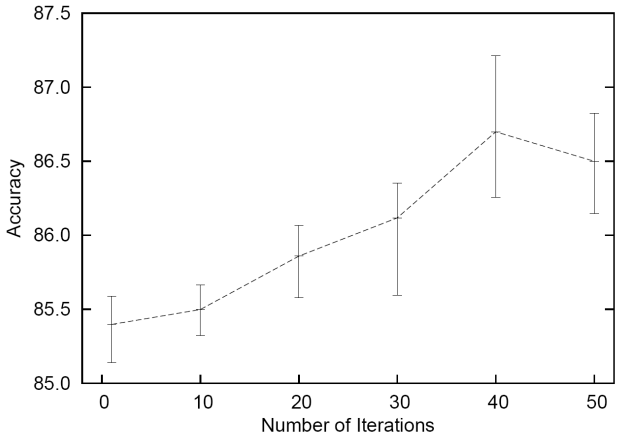

Figure 3. Average accuracies obtained from varying the *nIters* parameter and setting *nAnts* = 100, *nHops* = 12.

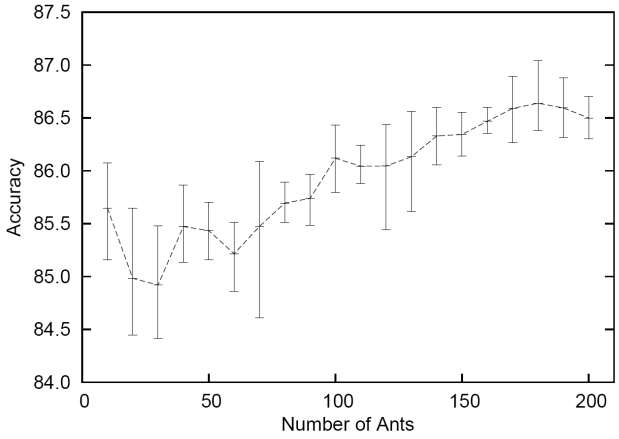

Figure 4. Average accuracies obtained from varying the *nAnts* parameter and setting *nIters* = 40, *nHops* = 12.

In Fig. 3, the model produces more accurate results when increasing the number of iterations. The reason may be that large number of iterations can increase opportunities for ants to repeatedly walk through some high quality paths. However, the accuracy is unexpectedly dropped after 40 iterations.

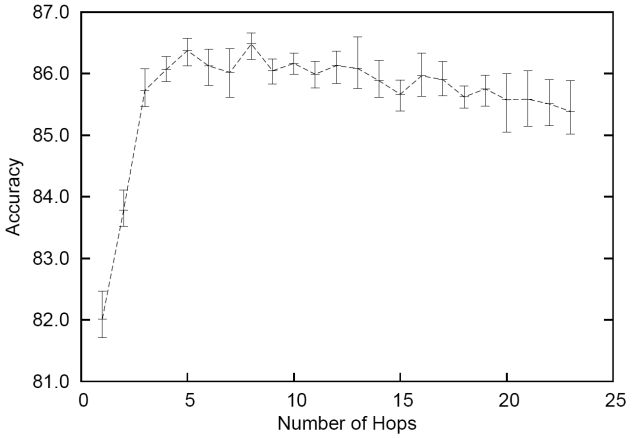

Figure 5. Average accuracies obtained from varying the *nHops* parameter and setting *nIters* = 40, *nAnts* = 100.

This phenomenon can be occurred since by nature the ACO-based learning is a randomized algorithm. Thus, ants can possibly select a wrong way. Similarly to the results depicted in Fig. 4, increasing the number of ants also tends to increase accuracy, due to several ants have more opportunities to discover several high quality paths.

In contrast to the results depicted in Fig. 5, increasing the number of hops tends to slightly decrease performance of the model. The reason may be that too long paths produce too general rules (i.e., having too many non-overlapped feature values) that cannot be used in classification; or, longer paths possibly contain more spam hosts that distort quality of rules.

V. Conclusion

In this paper, we propose LSD-ACO learning algorithm that follows the TrustRank assumption in order to generate classification rules for detecting spam. The experiments, conducting on the WEBSPAM-UK2006 collection, show that the LSD-ACO algorithm outperforms the other four baselines in all cases. From the sensitivity analysis, we found that more repeatedly iterative learning and employing more number of ants in the model can produce better results. However, walking too long distance may slightly drop the performance. In future work, we are looking forward to improving the model by giving a negative clue when discovering a wrong way, and hope to obtain higher quality set of classification rules.

Acknowledgment

This research is done under the support from Department of Computer Engineering, Faculty of Engineering, Kasetsart University. We thank to all research teams who help providing the experimental dataset. We also thank to anonymous reviewers for their comments and suggestions.

References

[1] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates, "Link-based characterization and detection of web spam," Proc. of the 2nd International Workshop on Adversarial Information Retrieval on the Web, 2006.

[2] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna, "A reference collection for web spam," ACM SIGIR Forum, vol. 40, no. 2, 2006, pp. 11–24.

[3] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri, "Know your neighbors: Web spam detection using the web topology," Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007.

[4] Y. Chung, M. Toyoda, and M. Kitsuregawa, "A study of link farm distribution and evolution using a time series of web snapshots," Proc. of the 5th International Workshop on Adversarial Information Retrieval on the Web, 2009.

[5] N. Dai, B. D. Davison, and X. Qi, "Looking into the past to better classify web spam," Proc. of the 5th International Workshop on Adversarial Information Retrieval on the Web, 2009.

[6] B. D. Davison, "Recognizing nepotistic links on the web," Proc. of AAAI Workshop on Artificial Intelligence for Web Search, 2000.

[7] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, 1999, pp. 137–172.

[8] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, 1997, pp. 53–66.

[9] M. Dorigo, V. Maniezzo, and A. Coloni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, no. 1, 1996, pp. 29–41.

[10] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," Proc. of the 13th International Joint Conference on Artificial Intelligence, 1993.

[11] D. Fetterly, M. Manasse, and M. Najork, "Spam, dam spam, and statistics: Using statistical analysis to locate spam web pages," Proc. of the 7th International Workshop on the Web and Databases, 2004.

[12] Z. Gyöngyi and H. Garcia-Molina, "Web spam taxonomy," Proc. of the 1st International Workshop on Adversarial Information Retrieval on the Web, 2005.

[13] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with TrustRank," Proc. of the 30th International Conference on Very Large Data Bases, 2004.

[14] M. R. Henzinger, R. Motwani, and C. Silverstein, "Challenges in web search engines," ACM SIGIR Forum, vol. 36, no. 2, 2002, pp. 11–22.

[15] Internet Archive, The wayback machine, `http://www.archive.org/`

[16] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," Journal of the ACM, vol. 46, no. 5, 1999, pp. 604–632.

[17] V. Krishnan and R. Raj, "Web spam detection with anti-trust rank," Proc. of the 2nd International Workshop on Adversarial Information Retrieval on the Web, 2006.

[18] J. Martinez-Romo and L. Araujo, "Web spam identification through language model analysis," Proc. of the 5th International Workshop on Adversarial Information Retrieval on the Web, 2009.

[19] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," Proc. of the 15th International World Wide Web Conference, 2006.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Technical Report, Stanford InfoLab, 1999.

[21] 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, 2002, pp. 321–332.

[22] I. H. Witten and E. Frank, "Data mining: Practical machine learning tools and techniques with Java implementations," Morgan Kaufmann, 2nd ed., 2005.

[23] B. Wu and B. D. Davison, "Identifying link farm spam pages," Proc. of the 14th International World Wide Web Conference, 2005.