Multimed Tools Appl (2018) 77:13249–13278
DOI 10.1007/s11042-017-4944-y

Distributed classification for image spam detection

Amiza Amir[1] · Bala Srinivasan[2] · Asad I. Khan[2]

Received: 30 June 2016 / Revised: 6 March 2017 / Accepted: 13 June 2017 /
Published online: 1 July 2017

Abstract Spam appears in various forms and the current trend in spamming is moving towards multimedia spam objects. Image spam is a new type of spam attacks which attempts to bypass the spam filters that mostly text-based. Spamming attacks the users in many ways and these are usually countered by having a server to filter the spammers. This paper provides a fully-distributed pattern recognition system within P2P networks using the distributed associative memory tree (DASMET) algorithm to detect spam which is cost-efficient and not prone to a single point of failure, unlike the server-based systems. This algorithm is scalable for large and frequently updated data sets, and specifically designed for data sets that consist of similar occurring patterns. We have evaluated our system against centralised state-of-the-art algorithms (NN, *k*-NN, naive Bayes, BPNN and RBFN) and distributed P2P-based algorithms (Ivote-DPV, ensemble *k*-NN, ensemble naive Bayes, and P2P-GN). The experimental results show that our method is highly accurate with a 98 to 99% accuracy rate, and incurs a small number of messages—in the best-case, it requires only two messages per recall test. In summary, our experimental results show that the DASMET performs best with a relatively small amount of resources for the spam detection compared to other distributed methods.

Keywords P2P classification · Distributed pattern recognition · Spam detection · Image spam · Distributed classification · Distributed data mining · P2P data mining

✉ Amiza Amir
amizaamir@unimap.edu.my

Bala Srinivasan
srini@monash.edu

Asad I. Khan
asad@monash.edu

[1] School of Computer and Communication Engineering, Universiti Malaysia Perlis, Perlis, Malaysia

[2] Faculty of Information Technology, Monash University, Melbourne, Australia

1 Introduction

Spam has been described in previous studies [4, 13] as one of the major cyber security problems. In email and messenger applications, spammers also use images to make such spam harder to detect than the more common text spam [11]. In messenger applications, spam may appear in the form of fake contacts which continuously send advertisements. In the peer-to-peer applications, spamming is notoriously used as a tool to control copyright infringement in P2P systems by the content owner, where it appears in the form of malware or polluted content which floods the system with bogus contents [35]. Recently, audio and video P2P streaming applications are also spammed with corrupted streaming chunks, which alter or disrupt the streaming service [31]. Given that spam has appeared in a variety of forms, spam filtering requires significant efforts to learn the complex patterns in order to classify these new forms effectively.

A few examples of email filtering techniques include greylisting, spamtraps, Domain Name System (DNS)-based blackhole lists, and content-based email filtering. In greylisting, a mail transfer agent temporarily rejects any email from an unknown sender and accepts the email when it is found legitimate. Spamtraps are email addresses that are used to trap email spam. Incoming email for other email addresses which have the same content with spam identified by spamtraps is blocked, and the IP address of the source is also stored to blacklist the sender. A DNS-based blackhole list is used to store the addresses of computers which involves in spamming and this list can be used by DNS server software later to query spammer address in real time. The success of this method, however, depends on the number of reports from users and the accuracy of the reports. Despite this, the most popular technique these days is content filtering where intelligent content recognition algorithms are used to predict the legitimacy of an email [6, 28].

The spam detection problem can be defined as follows. Given an object x, the spam detector must classify whether the object is a spam or a ham based on a number of features.[1] In pattern recognition, a supervised classifier is trained on a set of examples which consists of labelled spam and ham (legitimate) messages. This process generates a classifier model which can be used to predict either a message is a spam or not. In the text-based spam filtering, the message is transformed into a representation of d-dimensional vector where each dimension usually represents a character of words which is useful to distinguish spam. While in the image spam problem, the well-known method for generating these features is the content-based feature extraction [33]. The spam email or spam message domain usually involves a big and rich content that results to a large dimensional attribute vector. New spam objects are frequently introduced into the system and this results in a high volume amount of data for spam filters to deal with [3]. In addition, the pattern recognition techniques are usually computational expensive and hence to apply them on such big data on a single server is expensive. Therefore, we propose a fully-distributed computing approach for pattern recognition within P2P networks, namely Distributed Associative Memory Tree (DASMET) to solve this problem. The propose approach is an incremental learning algorithm which works efficiently with large and frequently updated data in spam problems. The incremental learning keeps the classifier constantly updated and provides the prediction

[1] The term feature is used alternately with the term attribute in this paper.

outcome at any time. By using DASMET in the proposed image spam detection technique, can provide an accurate, fast, scalable and economical option for this problem and this is shown in this paper.

Initially, we discuss the related works and then we explore the effectiveness and efficiency of the P2P-GN [1] for the spam detection problems. Since such problems involve a large number of duplicate or near-duplicate (almost similar) objects [26], we suggest that a more efficient (time and communication) solution can be achieved in classifying these objects. This is accomplished by extending the flat structure of the P2P-GN into a tree structure distributed associative memory as previously proposed in [2]. In the P2P-GN, the fine-grained memories of trained patterns are stored at leaf nodes and the association of these memories enables it to recognise fuzzy patterns. We extend this basic P2P-GN functionality by storing the memories in several different levels of view—in a tree structure where a node at the higher level of the tree stores a larger patterns' view (more coarse-grained memories) and the root node stores the whole patterns' view. Hence, the duplicate patterns can be detected in a straightforward way by the root, while the noisy patterns can be recalled by the nodes at the lower part of the tree (which stores fine-grained memories).

This comes with a trade-off of more storage overhead compared to the P2P-GN. Nonetheless, this overhead can still be considered as low since k training objects (instances) with exactly similar features (duplicate patterns) are only stored once—these are considered as a single unique pattern. Therefore, the trade-off can be ignored here; instead, we focus on improving the response time and communication overhead. The resulting extension of the P2P-GN is called the DASMET. In this paper, we study the feasibility of the DASMET to provide a fast, communication-efficient and fully-distributed solution for the image spam problems.

This remainder of this paper consists of seven sections as follows. Section 2 discusses the related works. The inefficiency of the P2P-GN approach for classification problems with a high number of repeating patterns, which led to the development of the DASMET for spam detections, are briefly explained in Section 3. In Section 4, we introduce the DASMET and describe its learning and recall[2] operations. Following this, we report the outcome of our experimental study of the DASMET's effectiveness and efficiency for distributed image spam detection in Section 6. Next, in Section 7, we analyse the complexity of the proposed scheme theoretically. Finally, we summarise the findings of this paper in Section 8.

2 Related works

Spam is often generated by machines using a common template and are sent in bulk to many users. Hence, these forms of spam are usually similar or exact duplicates. The duplicate spam items can easily be filtered out using a signature-based approach, where the signatures of the spam items are stored in a centralised database (e.g. *Sig2dat* [30]). This, however is infeasible for large data with rich and complex contents. Moreover, the signature-based methods are also unable to recognise the fuzzy spam items and, therefore, a degree of noise is often inserted by spammers to avoid detection that uses signature-based methods.

[2]Recall refers to classification or prediction in this context.

Distributed detection approaches have been widely used either to build a scalable method for large data or to solve the spam detection method within distributed systems (e.g. P2P systems). Zhou et al. [34] proposed a distributed spam detection system that uses feature signatures to identify spam emails and it can recognise fuzzy and "never-been-seen" spam items.The reputation-based approaches [17, 20, 23] also offer distributed approach for spam filtering. However, the high volume of spam objects results in a high communication overhead since votes for each object are exchanged among peers. In Zhou et al. [34], the identifiers for all objects which have one or more features that match with features of a queried object (test instance) are sent to the requester peer (the peer which requests for classification). In addition, the reputation-based methods and the method in Zhou et al. [34] require the unique identifiers (e.g. signatures) of all identified spam objects to be stored. This results in a high storage overhead since a botnet (a program to generate spam automatically) can easily generate a high volume of spam objects.

Several studies have proposed various methods based on pattern recognition to identify image spam. SpamHunter [14] proposed probabilistic decision tree by using color histogram and gradient orientation histograms for spam detection. Chen et al. [10] used STRHOG methods for intelligent character recognition on cloud enviroment. This paper focusses on the feature extraction part and it used HOG feature vectors and nearest neighbours classifier. Chowdhury et al. [11] proposed a spam filter by using BPNN classifier. In the method, file features of the embedded image and the low level; visual feature points are extracted. Image spam detection also has been introduced by Wakade et al. [32]. They proposed a spam filter by using J48 algorithms and six visual features are used: luminance, numbers of colors, color saturation, white pixel concentration, standard deviation of colors and hue.

The results from these studies show that pattern recognition techniques have proven effective for spam detections [25, 29] and these do not require the storing of the identifiers of all identified spam items as in Zhou et al. [34]. However, most of the available pattern recognition algorithms are centralised and given the spam objects are rapidly updated and involve a large amount of data, these algorithms are infeasible. Therefore, this paper investigates an efficient (time and communication), fully-distributed pattern recognition algorithm to detect spam within large data sets.

3 Efficient classification for datasets with frequent repeating patterns

In the P2P-GN, the memory of a pattern $\vec{x}$ is divided into n_H small parts and these parts are stored at n_H leaf nodes. To recall the memory of $\vec{x}$, a requester sends recall queries to all n_H leaf nodes and these leaf nodes respond with their recall predictions. The recall operation of a clean copy of a stored pattern $\vec{x}$ is similar to the recall of a noisy copy of $\vec{x}$. Hence, the recall process for both the clean and noisy copy of the stored $\vec{x}$ incur similar run-times and communication overheads. This results in inefficiency in the recall of duplicates of the stored patterns.

Let say the recall requests for a duplicate of x occurs at z times, then the total number of messages equals $2n_H \cdot z$. The number of messages can be reduced to $2z$ by storing the whole memory of x at a single node p_a where a recall request for a duplicate of x is sent to a single node p_a, instead of n_H leaf nodes. Meanwhile, a fuzzy pattern can still be predicted by recalling the memories at leaf nodes. This idea leads to the development of an extension

of the P2P-GN for datasets with a high number of repeating and similar patterns, that is the DASMET.

DASMET stores patterns in a tree structure; the leaf nodes store the fine-grained memory, and the nodes at the upper layer store a more coarse-grained memory where the top node has an overall view of the pattern. To recall a pattern, the algorithm initially detects the pattern at the root. If the pattern has been found, then the recall process is stopped here. Otherwise, we continue to look at a more close-up (fine-grained) view of the pattern at the lower-level nodes recursively until we discover the most likely predictions. By doing this, we can reduce the communication and processing overheads by detecting duplicates or closely-resembling patterns at the entry point of the system and thus avoid wasting resources on an expensive pattern recognition process. This is achievable because a high number of exact duplicate and similar patterns in the system is likely.

The brief idea behind our method is shown in Fig. 1. The example shows that the queried pattern is first matched with the stored pattern at level 0; if it is found to match the stored pattern, then the detection process decides that the label of the queried pattern equals the matched stored pattern. If it is not found, then the queried pattern is split into two smaller parts and these parts are matched with the equivalent parts of the stored pattern at level 1. For example, the region specified by the red line in the queried pattern is matched with the region covered by the red line ins the stored pattern. This process is continued until one of the matched region are found; this requires the memory of the whole segment and its parts. Instead of storing the pattern in its raw form, we only store the key of the pattern (pattern identifier). The number of iterations (levels) is also limited by the height of the tree since the smallest sub-pattern size is predefined (see Rule 1 of Section 4.1).

4 DASMET for pattern recognition within P2P networks

The DASMET structure was firstly introduced in [2]; however in this paper, we detail the processes involved. DASMET constructs the logical structure of the distributed algorithms, where every node in the tree represents a computational unit. Every peer within the network builds their own abstract DASMET tree structure using the parameters that are provided upon joining the system and all peers construct equivalent trees. The tree is used for reference in generating identifiers throughout the scheme's lifetime.

4.1 The logical structure of the DASMET

The logical structure of the DASMET is a rooted tree $D_R = (V_R, E_R)$, where V_R is the set of vertices and E_R is the set of edges (see Fig. 2). Each vertex stores a set of bias elements which include score tables. The distributed storage of the DASMET is the same as in the P2P-GN (as described in [1]) but with additional memories stored for the upper level of the tree . A requester node can be any peer within P2P networks and it does not store any memory, instead, it only requests for learning process or recall process and performs aggregations in calculating prediction for its own request. The DASMET structure depends on two predefined values: (i) the maximum number of children of each node, or the segment size at a leaf node, φ_s; and (ii) the number of features d. The DASMET operation starts by creating a number of sub-patterns or leaf nodes $\widehat{X} = \{\hat{x}_i\}_{i=1}^{n_h}$ prior to building the tree. The process of creating the sub-patterns or leaf nodes is similar to that of the P2P-GN. Using the

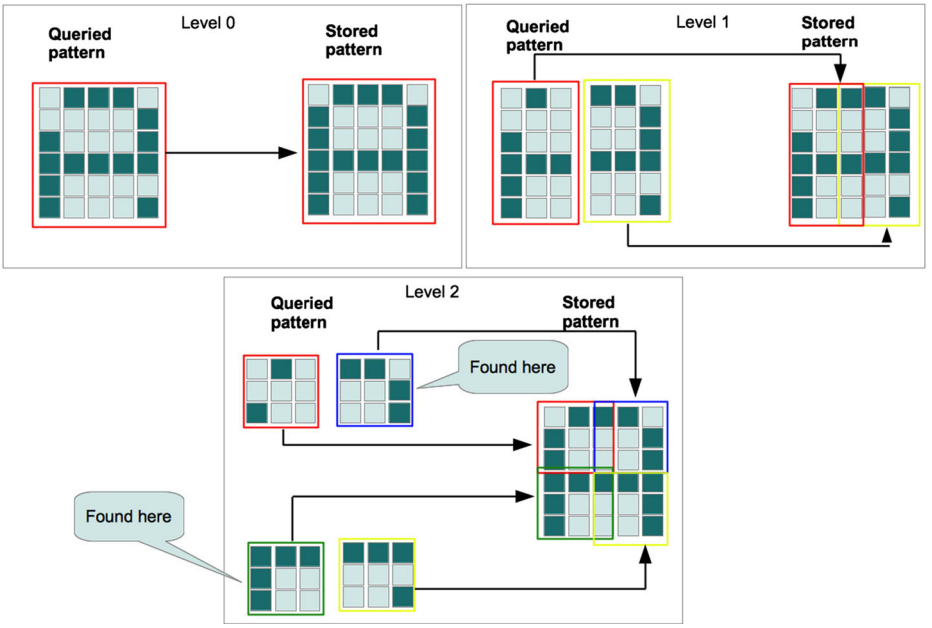


Fig. 1 An example of the idea used in the spam detection using DASMET. In level 0, the whole queried pattern is matched with the overall stored pattern and if it is not found, then in the second level, two parts of the pattern are matched against the stored pattern. The *coloured lines* show the matched pattern—for instance, the *blue region* of the queried pattern is matched with the *blue region* of the stored pattern. In this example, the similar patterns are found for two parts out of four at level 2

input $\widehat{X}$, the DASMET tree is generated by Algorithm 3 of Appendix A. Figure 2 shows an example of a DASMET structure for a problem where the number of features is 23. $\mathcal{SF}$ is segmented into 10 segments of size five each. The leaf nodes store the sub-patterns, while the internal nodes and the root node store the combined indices from its children. The tree height, $\hbar$, and the total number of nodes, n_R, are formally given in the following (1), and (2), respectively.

$$(log_{\varphi_s} n_H + 1) > \hbar \geq log_{\varphi_s} n_H \tag{1}$$

$$\begin{aligned} n_R &\leq \sum_{l=0}^{\hbar-1} \varphi_s^l + n_H \\ &\leq \frac{\varphi_s^{\hbar} - 1}{\varphi_s - 1} + n_H \end{aligned} \tag{2}$$

A memory of a unique pattern at a root node can be recalled by linking the entries within bias arrays at any level l. As shown in Fig. 2, the pattern "*amirakeepsthevaseformum*" can be recalled at any level by associating entries from nodes at the same level (see the dotted lines linking the entries from the bias arrays) as follows. At level 0, by associating the node h_{14} at entry 7. A level 1, by associating the entries: entry 5 at node h_{11} and entry 6 at node

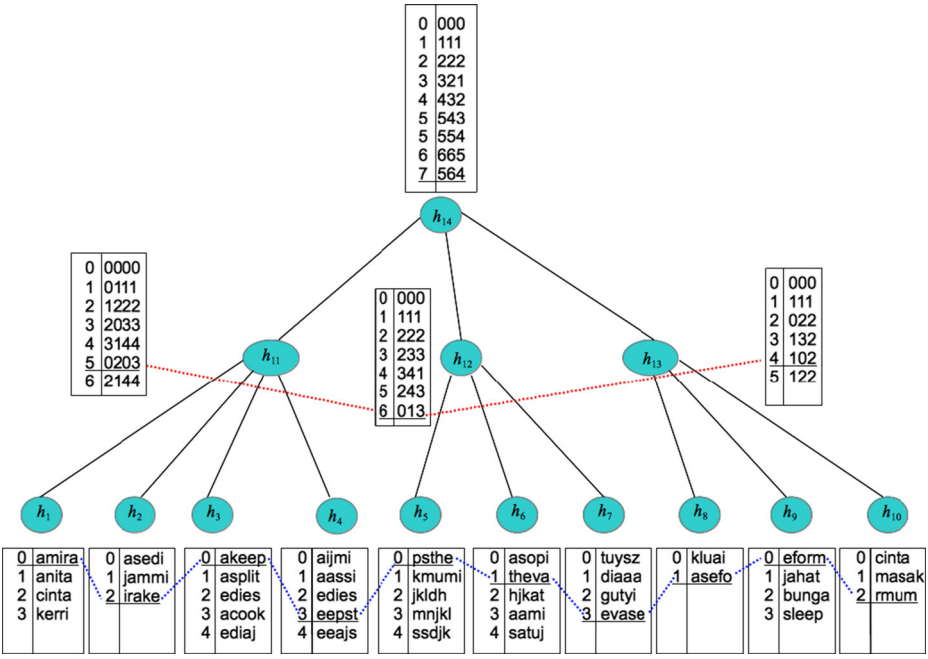


Fig. 2 An example of a DASMET architecture for a problem where d equals 23 and a newly added input pattern "*amirakeepsthevaseformum*". The predefined φ_s is 5. Each node holds a bias array (represented by the *box*). The *underlined entries* represent the active entries for the input pattern and the *dotted lines* show the links between the memories

h_{12} and entry 4 at node h_{13} and at level 2 by associating the entries: entry 0 at node h_1, entry 2 at node h_2, entry 0 at node h_3, entry 3 at node h_4, entry 0 at node h_5, entry 1 at node h_6, entry 3 at node h_7, entry 1 at node h_8, entry 0 at node h_9, and entry 2 at node h_{10}.

The selection of φ_s's value affects the performance of the algorithm and, therefore, a cross-validation may be required to make the decision. A large φ_s may reduce the accuracy of the DASMET by degrading the system's ability to recognise a highly fuzzy pattern. On the other hand, a small φ_s leads to an increase in the communication overhead and may result in the worst-case of recall run-time. For simplicity, we define a rule-of-thumb on φ_s and OV as follows.

Rule 1 (`DASMET Rule-of-Thumb`) *Given a pattern P with size-d, the value of φ_s is defined to be $3 \leq \varphi_s \leq \lfloor \log_2 d \rfloor$ with OV equal to $\lceil \frac{\varphi_s}{2} \rceil$ where $d > \varphi_s$.*

By limiting the maximum φ_s to $\lfloor \log_2 d \rfloor$, we can ensure the DASMET's capability to recognise a noisy pattern, while the minimum value of φ_s limits the communication overhead.

5 Network-wide DASMET

In the implementation within P2P networks, the bias array is divided into fine-granularity components where each component (represented by an identifier) includes a score table.

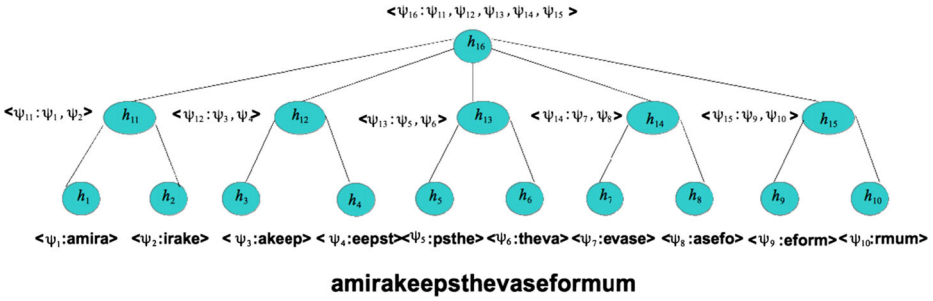


Fig. 3 Examples of pairs <bias identifier:input> that are created for the pattern "*amirakeepsthevaseformum*"

This forms a bias element which is similar to the storage in the P2P-GN [1]. A score table stores a list of pairs $\langle c, \widehat{f}(\psi, c)\rangle$ where class label is represented by c and the $\widehat{f}(\psi, c)$ is the frequency of pattern ψ occurring in class c. A bias identifier provides an address to every bias element across the network. The process to generate the bias identifier is explained as follows.

5.1 Bias identifier generation

Figure 3 shows an example of sub-patterns and bias identifiers created from the pattern "*amirakeepsthevaseformum*". In this example, 10 sub-patterns with size five each are created. The sub-patterns and their bias identifiers are shown in the form of pair <bias identifier : input>.[3] There are 14 bias identifiers in total where 10 bias identifiers represent the sub-patterns at level 2; three bias identifiers represent the sub-patterns at level 1; and one bias identifier represents the whole pattern at the level 0. If all of these bias identifiers have not yet been seen in the system (in the event that this is the first time the pattern "*amirakeepsthevaseformum*" occurs), then the learning process of these bias identifiers creates 14 new bias elements within the system (a bias identifier for each bias element).

The procedure for generating bias identifiers is described in Algorithm 4 of Appendix A. The bias identifiers for every node h_i in G_R are calculated from input $\vec{x}$ starting from level $\hbar$ until it reaches level 0. The generation of bias identifiers at leaf nodes (level $\hbar$) in the DASMET is similar to that of the P2P-GN, where the input to the identifier generation function qLeaf(.) is the sub-pattern and the position or index of the leaf node. However, an input argument to the identifier generation function q(.), for an entry at an internal/root node v, is an ordered set of combined identifiers from its child nodes $\widehat{\psi} = \{\psi_1, \psi_2, \cdots \psi_{w_v}\}$, where w_v is the number of child nodes and ψ_i is a bias identifier from ith child of v. Here, a hash function is used for the function q(.) and qLeaf(.).

Definition 2 (Function qLeaf(s, j)) An identifier generation function qLeaf(s, j) at a leaf node creates a bias identifier for a given input pattern s and position j.

[3]The input for generating a bias identifier at a leaf node is a raw sub-pattern, while the input at an internal node and the rootnode is a sequence of combined identifiers of its child nodes.

Definition 3 (Function $\mathsf{q}(\widehat{\psi})$) An identifier generation function $\mathsf{q}(\widehat{\psi})$ at an internal or root node creates a bias identifier for a given input of bias identifiers of its $|w|$ children $\widehat{\psi} = \{\psi_1, \psi_2, \cdots \psi_{|w|}\}$.

Any host is capable of calculating identifiers of a particular vector $\vec{x}$ by itself, provided that the G_R structure is known to all peers. Given the common parameters d and φ_s, a peer knows the G_R structure by building its abstract structure locally. This structure is used as a framework to create all bias identifiers of pattern $\vec{x}$.

In an example in Fig. 4, four bias elements and identifiers are created at the root node (i.e., $\{, \psi_{30}, \psi_{27}, \psi_{20}, \psi_{16}\}$). The bias identifiers at the root node are created using the combined identifiers from the leaf nodes {leaf node 1, leaf node 2, leaf node 3}. For instance, the bias identifier ψ_{30} (at the root node) is created using the combination of identifiers ψ_3 (in the leaf node 1), ψ_8 (in the leaf node 2) and ψ_{11} (in the leaf node 3) as input parameters, while a bias identifier for a bias element at a leaf node is generated by using an input sub-pattern.

In the context of P2P networks, an identifier or a key of a data must comply with the overlay network identifier space for efficient storing and lookup. The consistent hash function that creates the identifier is defined by the P2P overlay design and it is provided to the peer when it joins the network. In our implementation, we use SHA-1 to create the identifiers. In the placement within P2P networks as shown in Fig. 4, a bias element with identifier ψ is placed at a peer which has identifier $p.ID$ where the ψ is within the range of $[p.ID, p_{suc[0]}.ID)$, $p.ID$ is an identifier of p and $p_{suc[0]}.ID$ is an identifier of the immediate successor of p.

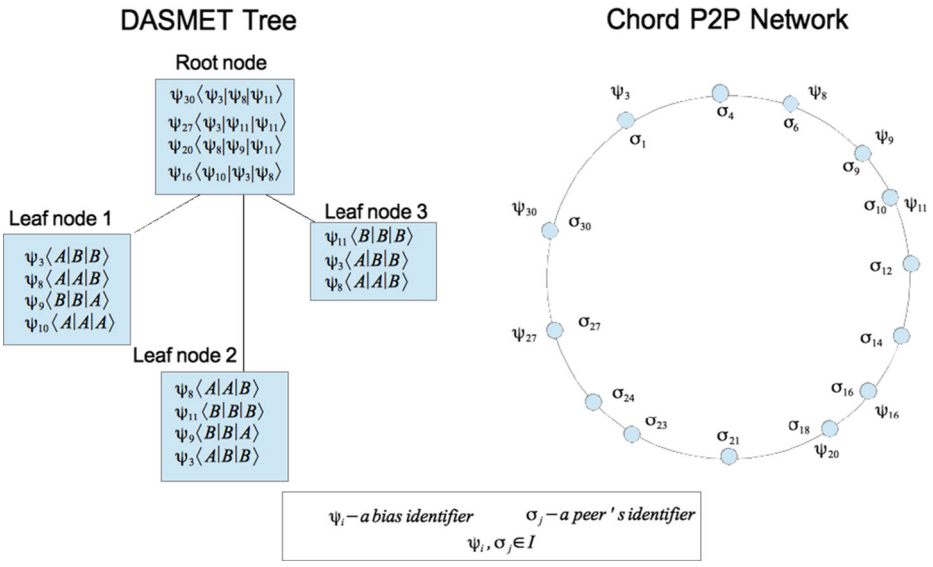


Fig. 4 The diagram on the *left* shows the logical location of bias arrays and bias elements within the DASMET structure, while the diagram on the *right* shows the placement of bias elements within a Chord network. Bias identifiers are generated in the same identifier space as peers' identifiers: a bias element with identifier ψ_8 being assigned to the peer σ_6 where the peer σ_6 is responsible for storing bias elements with identifiers within the range [6, 9)

Here, we use only sub-patterns as input for bias identifier generation. For instance, the leaf node 1, leaf node 2 and leaf node 3 have similar sub-patterns $\langle A|A|B\rangle$ and; therefore, the same identifiers ψ_8 are generated for these sub-patterns. However, henceforth in this paper, we use the pair of sub-pattern and position as input for bias identifier generation. For a pattern with the same domain values (e.g., {0, 1}), there is a high possibility that a segment $\hat{x}_i$ at the position i is similar to a segment $\hat{s}_j$ at the position j. Therefore, the use of only sub-patterns for identifier generation will result in a smaller number of bias elements compared to the use of the pairs of sub-pattern and position. However, this may also result in the peer which holds the bias elements to responding to a higher number of recall or learning requests.

5.2 DASMET learning procedure

The learning process in the DASMET strategy is explained in Algorithm 1.

Algorithm 1 DASMET learning

1: Input : c, $\vec{x}$, and H
2: **At requester peer x**
3: $\{\psi(h_1), \psi(h_2), \cdots \psi(h_{n_R})\} \leftarrow$ generatingIdentifier$(l, \vec{x})$.
4: **for all** $h_i \in G_R$ **do**
5: Sends LEARN_REQUEST for $\{\langle\psi(h_i), c\rangle$ to the network.
6: **end for**
7:
8: **At a peer y: upon receiving LEARN_REQUEST from peer x**
9: Perform localLearning(.).

The input to the DASMET learning is a set of leaf nodes H, and a training instance which consists of a label for the training instance c and a sequence of feature vectors $\vec{x}$ (which forms a pattern). In the DASMET learning, all the indices are calculated at a requester peer (see Section 5.1). In this instance, all peers are assumed to know the tree structures (which they can build themselves locally given the tree parameters) and the identifier generation function. This information can be shared during peers' bootstrapping. Therefore, all peers know $\hbar$, that is, the height of the DASMET tree. The localLearning(.) function and LEARN_REQUEST message here are equivalent to the P2P-GN.

The bias identifiers for all nodes in DASMET (G_R) and the class label for the training instance in pairs $\{\langle\psi(h_i), c\rangle\}_{h_i \in G_R}$ are sent to the network using the LEARN_REQUEST messages where $\psi(h_i)$ is the bias identifier that is generated for node $h_i \in G_R$ and c is the class label. These bias identifiers are generated using the generatingIdentifiers(.) function which has been discussed in the previous Section 5.1. Upon receiving the LEARN_REQUEST message, recipient peers then locally perform the local learning and the learning process.

5.3 DASMET recall procedure

The procedure during recall is explained in Algorithm 2. Three main functions—localRecall(.), agg(.), and finalPrediction (.)—and two types of messages—RECALL_REQUEST and RECALL_RESPONSE are the same functions as in the P2P-GN.

Algorithm 2 DASMET recall

At requester peer x
$\{\psi(h_1), \psi(h_2), \cdots \psi(h_{n_R})\} \leftarrow$ generatingIdentifier$(l, \vec{x})$.
$l \leftarrow 0$
$h_1 \leftarrow$ root node.
Sends RECALL_REQUEST for $\psi(h_1)$.

At a peer z: upon receiving RECALL_REQUEST $\psi(h)$ from a peer x
Calculate prediction $\hat{r}(h)$ using function localRecall$(\psi(h))$.
Send RECALL_RESPONSE with $\hat{r}(h)$ to the peer x.

At requester peer x: upon receiving responses $\hat{r}(h)$
if $l \neq 0$ **then**
    $w_x \leftarrow$ the number of children of the peer x.
    Aggregate the predictions to obtain a prediction $\widehat{R}(G)$ using agg$(\{\hat{r}(h)_i\}_{i=0}^{w_x})$
else
    $\widehat{R}(G) = \hat{r}(h)$
end if
if $C_0 == true$ **then**
    $\mathcal{L} =$finalPrediction$(\widehat{R}(G))$.
else
    $l++$
    $H_l \leftarrow$ is a set of nodes at level-l.
    for all $h_i \in H_l$ **do**
        Sends RECALL_REQUEST for $\psi(h_i)$.
    end for
end if

The recall is invoked by a requester peer and it starts from the root node at level $l = 0$ and after completing the generatingIdentifier(.) execution. The lookup begins by sending a query RECALL_REQUEST for identifier $\{\psi(h_1)\}$ to the network. If the receiving peer has the queried index, then it calculates the prediction using the local function localRecall$(\psi(h_1))$ and then responds with the RECALL_RESPONSE message with its local prediction $r(h_1)$. Upon receiving the response, the requester validates the prediction using Rules 4 and 5.

Rule 4 (Termination Condition C_0) *The termination condition, C_0, of this algorithm is true in the event of: (i) a single prediction is obtained or, (ii) there is no valid prediction made (by Rule 5) or, (iii) the global prediction for queries of indices for level $\hbar$ has been calculated.*

Rule 5 (Invalid Prediction Rule) *A prediction is invalid when it consists of more than one class which have a vote value that is equal to or larger than $max_v - v_e$ where max_v is the majority vote value and v_e is a discount vote value which is an integer $0 \leq v_e < max_v - 1$*

In the event the prediction is invalid ($\mathcal{C}_o$ of Rule 4 is false), the next recall phase is executed with recall requests for the next level being sent to the network, whereas if it is valid ($\mathcal{C}_o$ of Rule 4 is true), the recall process is terminated with $\hat{r}(h_1)$ as a global prediction $\widehat{R}(G)$.

As the global prediction is obtained at the first recall, then we achieve the optimal recall prediction (see Property 6). For every recall phase, $m_l = |H_l|$ requests for $\{\psi(h_i)\}_{h_i \in H_l}$ are submitted into the network where H_l is a set of nodes at level l and $H_l \in G_R$.

In the event that the prediction is invalid, the recipient peer z calculates the local prediction $\hat{r}(h)$ using the local function $\mathsf{localRecall}(\psi(h))$ and then replies to the requester x. After receiving the local predictions from m_l peers, the requester performs an aggregation function $\mathsf{agg}(\{r(h_i)\}_{h_i \in H_l})$ to finalise the global prediction $\widehat{R}(G)$. Then $\widehat{R}(G)$ is validated before terminating ($\mathcal{C}_o$ of Rule 4 is true) or proceeding ($\mathcal{C}_o$ of Rule 4 is false) with the lookup for the recall requests at the next level ($l = l+1$). These processes are stopped when $l > \hbar$. The final $\widehat{R}(G)$ is then used to obtain a final prediction $\mathcal{L}$ using the function $\mathsf{finalPrediction}(.)$. This is formally defined in Property 6.

Property 6 (`Optimal Recall Prediction`) *An optimal recall is achieved when the valid prediction is obtained during level 1 (the lookup query is found at the root node) which only requires a single query message that is sent to the root node and then the root node responds with a valid prediction.*

In the case of an optimal prediction with optimal connections,[4] the number of messages is two and the recall process requires a constant time.

6 Experiment

In this section, we empirically evaluate the accuracy and efficiency of our methods for image spam detection. All the experiments here were conducted using 10-fold cross-validation and they were run on a PC with Intel Core i-7 2.9GHz and 8Gb memory. All algorithms were implemented using Java and they were integrated into the Peersim simulator [27].

6.1 Dataset

The dataset that was used for the image spam experiment is a publicly available dataset and it was obtained from the personal image dataset Dredze et al. [12] and the Amsterdam library of object images (LOI) Geusebroek et al. [15]. We used the Weka package [18] to partition the training and test datasets for the 10-fold cross-validation in all experiments. The personal image dataset consists of 3293 images of spam and 1993 images of ham.

Due to difficulties during feature extraction (the feature extraction tool that we used in this experiment was unable to extract the required features), 63 images of spam and 121 images of ham have been removed. To increase the number of ham images, we added 2000 images of ham from the Amsterdam LOI corpus to increase the size of dataset for our experiments, resulting in a dataset of 3872 ham images and 3266 spam images.

The large number of ham images is useful for investigating the false positive predictions generated by our algorithm. This is important since the high number of false positives is a common problem of spam filtering—a high number of ham objects is misclassified as spam, even though they contain legitimate content. Figures 5 and 6 show a few examples of images that were used in this experiment.

[4] The lookup process only involves a single hop message.

Fig. 5 Some examples of image spam that are used in this experiment

6.2 Content-based feature extraction

A tool utilising the Java Advanced Image (JAI) package that is called JFeatureLib [21] was used in this experiment to extract the features. Using Weka, we performed discretisation to transform the continuous values in the dataset into discrete values since our algorithm works only on a discrete dataset.

Most of the research on image spam detection generates content features using Optical Character Recognition (OCR) techniques, since image spam commonly includes embedded texts. However, the embedded text is always changed to avoid detection. Therefore, in this experiment, we used low level features low level features (corner and edge detection, shape, colour and textures) which are better suited to this problem since the quality of images which are generated by machine are usually not as good as the images produced by humans Al-Duwairi et al. [5]. Here, we selected three low level descriptors—Haralick descriptor [19], colour and edge directivity descriptor (CEDD) [8], and fuzzy colour and texture histogram (FCTH) [9], due to their low computation overheads. The Haralick differentiates the image areas based on textural characteristics. It produces 14 texture features: angular second moment, contrast, correlation, variance, inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, information measures of correlation, information measures of correlation and maximum correlation coefficient [19]. The CEDD incorporates colour and texture information in histogram form and it produces 144 features, where each feature represents a bin [8]. The FCTH combines the colour and texture information in a single histogram which consists of a sequence of 192 bins which represents a feature vector [9].

Fig. 6 Some examples of image ham that are used in this experiment

6.3 The classifiers

We compare the accuracy of our algorithm for spam detection with a set of five centralised classifiers: k-nearest neighbour (k-NN), nearest neighbour (1-NN), naive Bayes, backpropagation neural network (BPNN) and Gaussian radial basis function network (RBFN). We further compare the accuracy of our algorithm with three distributed algorithms: Ivote-DPV [24], ensemble k-NN and ensemble naive Bayes. It is also important to highlight that almost all algorithms in this experiment—1-NN, k-NN, naive Bayes, ensemble k-NN, ensemble naive Bayes, DASMET and P2P-GN)—are update-able as they perform incremental learning. Hence, they are feasible for online learning which is useful for P2P networks in dealing with large datasets and frequent data updates. The experimental set-up of these classifiers is as follows.

Out of the five centralised classifiers, three of them require some parameter tuning. These parameter values are selected either by manual tuning or in consideration of the computational limitations of the hardware used in this experiment. Particularly in the case of BPNN, we set the learning iterations to 50, since it was expensive to process a large number of iterations. The number of hidden neurons in BPNN was set to $\frac{d+|\mathcal{C}|}{2}$ where d is the number of attributes (features) and $|\mathcal{C}|$ is the number of classes. For RBFN, the classifier uses the k-means clustering algorithm for basis functions and it learns a logistic regression. The number of iterations is set to be unlimited until convergence. For the k-NN classifier, we used $k = 3$ in this experiment.

In the Ivote-DPV experiments, we used a pasting algorithm—Ivote—as the local classifier and J48 as the base classifier (as used in Luo et al. [24]). In the Ivote, a small number of instances (called bites) are sampled from the local dataset during each learning iteration.

We set the size of bites to 20% of the dataset size when the dataset size is larger than or equal to 100. Otherwise, the size of bites is equal the size of local dataset size. This is to avoid very small bites sizes, which may result in a high number of iterations before convergence. Here, we used the same setting as in Luo et al. [24], where the parameter $\lambda = 0.002$ was given for the Ivote classifier. For the experiments of ensemble classifiers (ensemble k-NN and ensemble naive Bayes), the local results from all the peers were collected at a single peer and these were aggregated using majority voting.

6.4 The P2P-GN and DASMET structure

The P2P-GN and the DASMET structures are problem-specific. Since there are three different datasets created with three different descriptors in the image spam detection experiment (the Haralick descriptor, the CEDD descriptor and the FCTH descriptor), we have three different DASMET trees in this experiment.

In order to simplify this experiment, the structures of DASMET were determined based on Rule 1, which provides a simple guideline of parameter selection. The DASMET structures that we used in this experiment are described in Table 1. φ_s values were obtained by applying Rule 1 to each problem. For the Haralick experiment, we decided to use $\varphi_s = 3$ (that is the smallest value allowed by Rule 1) since the dataset has a very small number of features ($d = 14$). In contrast, we used the highest values allowed by Rule 1 ($\varphi_s = \lfloor \log_2 d \rfloor$) in the FCTH and CEDD experiments, in that d in both of these experiments is high. Note that the values were chosen using tuning-by-hand and based on the idea that a high φ_s may reduce accuracy, while a small φ_s increases communication overheads.

The parameters to create the sub-patterns in the P2P-GN are given as follows. The sub-pattern size, d_s is set to be equal to φ_s and the overlap rate is OV. The values of φ_s and OV for the experiments in this chapter are given in Table 1. Hence, the same number of sub-patterns or number of leaf nodes are generated in both, the P2P-GN and the DASMET in these experiments.

Cross-validation for parameter selection or parameter optimisation may be useful to get better results. Here, however, we do not optimise the parameters for this particular problem.

6.5 Simulator set-up for distributed algorithms

The experiments were conducted on a simulation of a 2000 peers Chord network [22] using the Peersim simulator. We chose this value due to the small size of the datasets in both experiments. Considering the small number of training instances in this experiment, a distributed system simulation of a network of 2000 peers may result in a very small number of training instances at each peer in a disjoint data distribution. Therefore, in the simulation

Table 1 The P2P-GN and DASMET structures for this experiment

Dataset	Data dimension (d)	Segment size/ max. children (φ_s)	Overlap rate (OV)	Number of leaf nodes (n_H)	Total nodes (n_R)	Height ($\hbar$)
Haralick	14	3	2	12	22	3
FCTH	192	7	4	63	85	3
CEDD	144	7	4	47	55	2

of these these classifiers, we distributed the training dataset using a joint data distribution where we set 50 instances per local dataset and the training instances in each local dataset are randomly sampled with replacement from the training dataset, using a uniform distribution. A joint data distribution from a small size training dataset within a large network size results in a large number of redundant samples across the network. Hence, the size of the local datasets and the number of peers are reasonable, considering the size of the datasets in this experiment.

Every peer in this simulation stores the link to its k successors and a predecessor. Therefore, every peer has $k + 1$ immediate neighbours. The delay per hop was uniformly distributed between 1 and 20 ms in this experiment. For Chord network, the size identifier is equal to 160-bit and the size of the successor list is equal to 12.

6.6 Performance metrics

All of these experiments involve binary classification: that is, classifying an item of interest into either spam or ham (legitimate object or non-spam) classes. We used five metrics to evaluate accuracy in these experiments: *accuracy*, *false positive rate (FP rate)*, *precision*, *true positive rate (TP rate)*[5] and the *F-measure* (F_m)[6] [7]. To evaluate the overheads of the distributed algorithms, two metrics which are affected by the networking related factor were used: number of messages per test and the recall time per test. These performance metrics measure the communication efficiency and time efficiency of the distributed algorithms, particularly for the fully-distributed image spam detection problem.

The accuracy is calculated by dividing the correctly classified tests with the total tests. The FP rate measures the rate of ham objects which are incorrectly classified as spam. This measurement is important since the problem with the current spam filters is the high false positive rate, which incorrectly filters out the ham objects from the user's view. The precision measures the fraction of the queries that are correctly recalled as spam from all the queries that are classified as spam. The precision is always used together with the TP rate metric which gives the fraction of queries that are correctly recalled as spam from all the queries which are actually spam. Finally, F_m is the harmonic mean of precision and TP rate (a balanced mean between precision and TP rate).

6.7 Evaluating accuracy

The accuracy results for the FCTH experiment, CEDD experiment and Haralick experiment are reported in Tables 2, 3 and 4, respectively. In general, the result shows that 1-NN performs the best among the centralised classifiers (k-NN, BPNN, RBFN and naive Bayes) in these experiments. It has very high accuracy, a low FP rate and a high F_m rate. It generally has the lowest FP rate compared to other centralised and distributed classifiers except in the FCTH experiment.

Within the group of distributed classifiers, the DASMET significantly outperforms other methods in accuracy. The P2P-GN is less accurate than the DASMET but both are better than the ensemble k-NN, ensemble naive Bayes and Ivote-DPV. The Ivote-DPV performs poorly compared to all other classifiers where its FP rate for all experiments is the highest and its accuracy rate is the lowest.

[5]True positive rate is equivalent to recall in the content retrieval context. However, we do not use the term recall here since, in the context of this paper, recall refers to prediction.

[6]F-measure is also known as F-score or F_1 score.

Table 2 Accuracy in the experiment using the FCTH descriptor for image spam detection

Classifier	FP rate	Accuracy	Precision	TP rate	F_m
Centralised classifiers					
Naive Bayes	0.107	0.871	0.877	0.846	0.861
1-NN	0.015	0.971	0.983	0.955	0.969
k-NN	0.014	0.959	0.984	0.930	0.956
BPNN	0.029	0.968	0.966	0.965	0.966
RBFN	0.105	0.897	0.873	0.898	0.885
Distributed classifiers					
P2P-GN	0.069	0.941	0.915	0.954	0.934
DASMET	0.018	0.972	0.979	0.960	0.970
Ensemble *k*-NN	0.138	0.857	0.834	0.851	0.842
Ensemble naive baye	0.125	0.857	0.855	0.836	0.845
Ivote-DPV	0.152	0.822	0.825	0.795	0.810

The DASMET generally has the best accuracy compared to other classifiers in all experiments. It has accuracy comparable to the centralised 1-NN where its F_m in all experiments are higher than 1-NN, although its FP rate is slightly higher than 1-NN. It achieves 0.97 to 0.987 accuracy rate with 0.97 to 0.986 F_m value. The high F_m value demonstrates the consistency of the predictions. The low FP rate—less than 0.02 for all the experiments using different descriptors—show that DASMET is a good filter since it rarely classifies ham objects as spam, such cases of misclassification often hinder the users viewing the legitimate images which is possibly important or useful to them.

Since the centralised classifiers have all datasets at a single node (where the learning and classification are executed at a single site), this gives an advantage for them to produce a more accurate prediction. The comparison, however, aims to show that the proposed distributed algorithm can produce a comparable accuracy to the centralised implementation

Table 3 Accuracy in the experiment using the CEDD descriptor for image spam detection

Classifier	FP rate	Accuracy	Precision	TP rate	F_m
Centralised classifiers					
Naive Bayes	0.116	0.862	0.883	0.827	0.855
1-NN	0.009	0.972	0.990	0.949	0.969
k-NN	0.013	0.955	0.987	0.920	0.952
BPNN	0.032	0.972	0.968	0.969	0.969
RBFN	0.111	0.896	0.890	0.883	0.886
Distributed classifiers					
P2P-GN	0.045	0.960	0.946	0.967	0.956
DASMET	0.014	0.983	0.984	0.978	0.981
Ensemble *k*-NN	0.120	0.827	0.873	0.777	0.822
Ensemble naive Bayes	0.115	0.833	0.878	0.783	0.828
Ivote-DPV	0.152	0.820	0.827	0.790	0.808

Table 4 Accuracy in the experiment using the Haralick descriptor for image spam detection

Classifier	FP rate	Accuracy	Precision	TP rate	F_m
Centralised classifiers					
Naive Bayes	0.049	0.951	0.942	0.951	0.946
1-NN	0.005	0.985	0.994	0.973	0.983
k-NN	0.010	0.975	0.989	0.959	0.974
BPNN	0.021	0.978	0.975	0.978	0.976
RBFN	0.048	0.948	0.943	0.943	0.943
Distributed classifiers					
P2P-GN	0.015	0.984	0.982	0.982	0.982
DASMET	0.009	0.987	0.989	0.982	0.986
Ensemble *k*-NN	0.106	0.911	0.867	0.933	0.899
Ensemble naive Bayes	0.087	0.920	0.894	0.927	0.910
Ivote-DPV	0.164	0.846	0.794	0.859	0.825

of the state-of-the-art algorithms. Nonetheless, since DASMET has shown higher F_m than the state-of-the-art centralised algorithms (1-NN, *k*-NN, naive Bayes, BPNN and RBFN) for the image spam detection problem, we have proven that our fully-distributed algorithm is indeed competitive with these well-known centralised classifiers. We also found that the Haralick descriptor gives better accuracy for image spam detection than the other two descriptors. This shows the efficiency and effectiveness of the Haralick descriptor for this problem, considering that it is computationally cheaper and produces a smaller number of features compared to the other two methods.

6.8 Evaluating communication overheads

The number of messages exchanged per test in the ensemble *k*-NN and ensemble naive Bayes is clearly high (linear in network size), with an equivalent magnitude in all experiments. This is because all peers send their local results to a single requester peer in these algorithms. For that reason, we only report the communication overheads for the P2P-GN, DASMET and the Ivote-DPV in this section.

Since the learning and classification process in the P2P-GN and the DASMET involve finding a key within a DHT system, their communication overheads are influenced by the lookup overhead of the underlying P2P system—the Chord network in this experiment. In contrast, the number of messages per test in the Ivote-DPV relies upon the number of neighbours per peer, since a peer communicates with its immediate neighbours during aggregation. In all experiments, we observed that the distribution of the number of messages per test of the P2P-GN has a much lower standard deviation compared to the DASMET and the Ivote-DPV which are spuriously distributed.

The number of messages per test in the image spam experiment are reported in three histograms: Fig. 7 for the experiment using the FCTH feature set, Fig. 8 for the experiment using the CEDD feature set and Fig. 9 for the experiment using the Haralick feature set. Each of the experiments involves a total of 7138 tests from the 10-fold cross-validation.

In all these experiments, the number of messages per test ranges from 13 messages to 2000 messages in the Ivote-DPV. The high variance is due to two reasons. First, when there

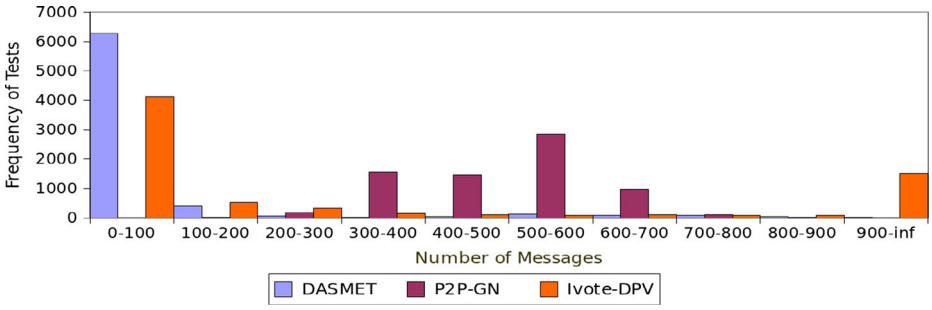


Fig. 7 Number of messages per test in the FCTH experiment

is no conflict during classification (every peer agrees to a single decision), the number of messages per test is minimal; it is equal to the number of neighbours ($k + 1$ where k is the size of the successors list). In this case, a requester peer sends its prediction to its neighbours and if the neighbours also predict similarly, then none of them respond to the requester; hence the DPV process stops within one iteration. Second, when there are high conflicts (high disagreement on the decisions) among peers, in the worst case, the number of DPV iterations equals the network size.

Figure 9 depicts that in all tests the DASMET used fewer than 200 messages in the experiments using the Haralick descriptors. In the experiment using the FCTH descriptor (see Fig. 7), the number of messages per test in the P2P-GN is normally distributed within a range of 100 to 900 messages, while in the experiment using the CEDD (see Fig. 8), the number of messages per test is normally distributed between 100 and 700.

The DASMET tests show that the lowest number of messages per test is three in all experiments, while the highest number of messages per test is 715 in the CEDD experiments, 1105 in the FCTH experiments and 242 in the Haralick experiments. A large proportion of the tests (approximately 87%) consumed less than 100 messages and all of the P2P-GN tests in the CEDD and FCTH experiments required at least 100 messages. Apart from the FCTH experiments, where 27 out of 7138 tests used more than or equal to 900 messages, none of the DASMET tests in the CEDD and Haralick experiments required 900 or more messages. This shows the efficiency of the DASMET which provides highly accurate prediction despite consuming a low amount of resources.

Due to the small number of features, the communication overheads in the P2P-GN and the DASMET are smaller when using the Haralick descriptors compared to the CEDD and

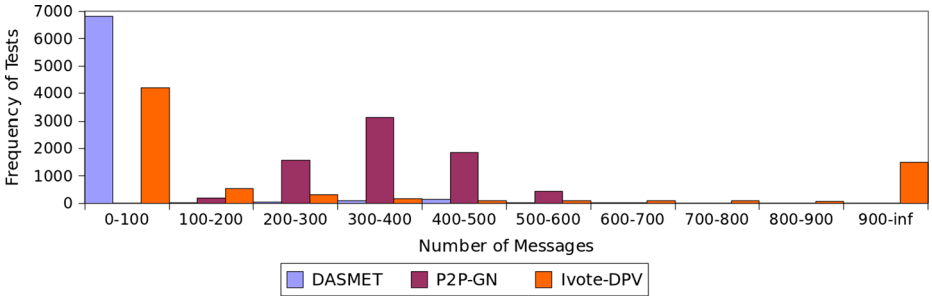


Fig. 8 Number of messages per test in the CEDD experiment

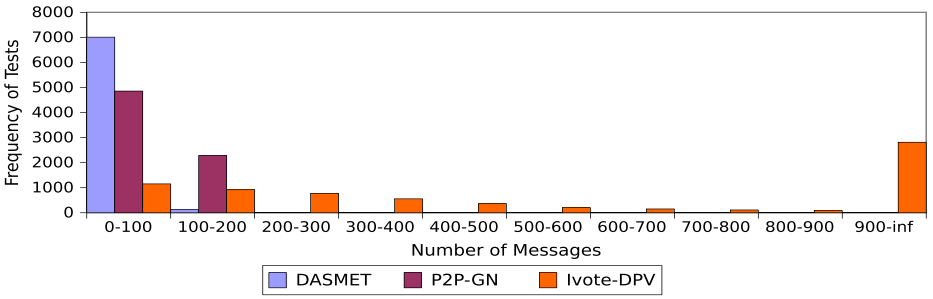


Fig. 9 Number of messages per test in the Haralick experiment

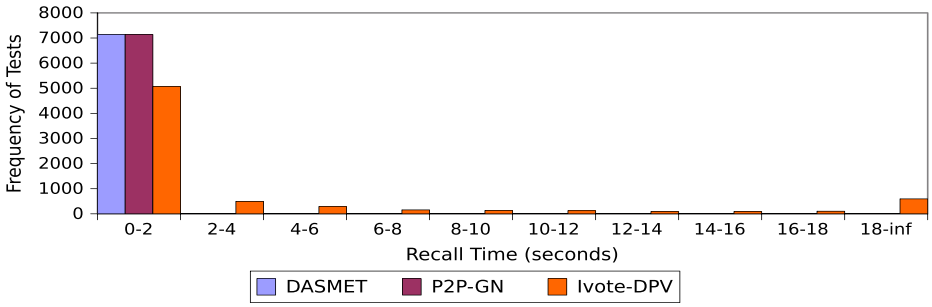
FCTH descriptors. The experiment using the Ivote-DPV, however, shows differently, as the number of messages per test are found to be greater, with nearly 40% of the tests using at least 900 messages. This suggests that the low number of features may be attributed to the increase in conflicts in the Ivote-DPV classification process, which results in an increase in communication overheads.

6.9 Evaluating recall time

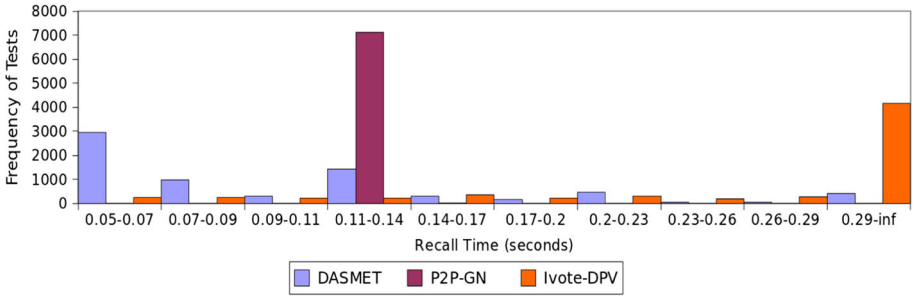
In this section, we report the recall time per test in the image spam detection (based on 7138 observations). The number of observations is obtained from all the 10-fold cross-validation experiments. Due to the sparseness of recall time distribution, particularly in the Ivote-DPV and the P2P-GN tests, the recall time per test is reported in two different histograms: *Histogram 1* with the high range of 0 to 18 s (as shown in Figs. 10a, 11a and 12a) and *Histogram 2*, with the short range of 0 to 0.29 s (as shown in Figs. 10b, 11b and 12b). Histogram 2 is further used to analyse the comparison of recall time per test between the DASMET and the P2P-GN, since all the tests in these two algorithms were completed within less than 1 second. In the DASMET tests, we found that fewer than 1% of tests in the image spam detection experiment took 0.29 s or more recall time. However, all the P2P-GN tests were completed within less than 0.29 s.

In Histogram 1 as presented in Figs. 10a, 11a and 12a, the results show a similar trend in the FCTH, CEDD and Haralick experiments, respectively. We found that all the DASMET tests and all the P2P-GN tests; and about 70 to 72% of the Ivote-DPV tests used less than 2 s of recall time. Nonetheless, around 7 to 8% of tests from the Ivote-DPV experiment consumed at least 18 s of recall time. Additionally, the highest recall time per test from these experiments was found in the Haralick experiment applying Ivote-DPV, with 47.36 s. The reports from Histogram 2 (see Figs. 10b, 11b and 12b) depict about 5.8% of DASMET tests in the FCTH experiment and less than 1% of the DASMET tests in the CEDD and Haralick experiments as requiring at least 0.29 s. In the experiments applying the P2P-GN, all of the tests were completed in less than 0.17 s. In contrast, about 57 to 60% from the Ivote-DPV tests required at least 0.29 s of recall time. This suggests that the DASMET and the P2P-GN are more suitable than the Ivote-DPV for an application which requires a fast response.

Every test in the experiment using either the CEDD or FCTH descriptors and applying the P2P-GN completed within 0.09 s to less than 0.17 s. In the experiment using the Haralick descriptor, the P2P-GN required within 0.07 s to less than 0.17 s to process each test. In other words, 100% of the P2P-GN tests were completed in less than 0.17 s. However, in



(a) Histogram 1 for recall time per test on the FCTH feature set



(b) Histogram 2 for recall time test on the FCTH feature set

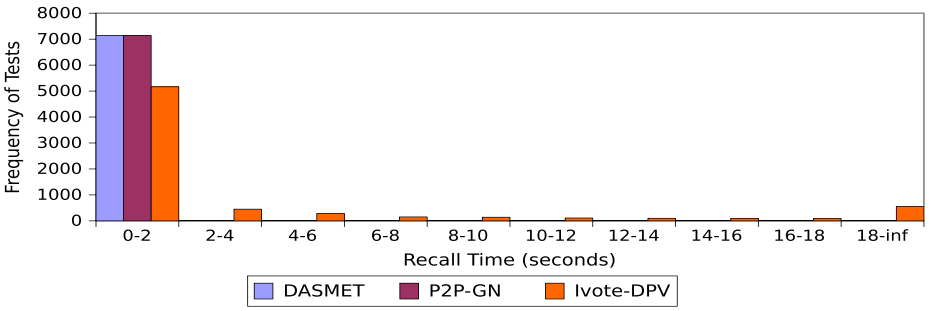
Fig. 10 Recall time per test in the FCTH experiment

the CEDD, Haralick and FCTH experiments, the percentage of the DASMET tests that consumed less than 0.17 s of processing time is 89.5, 88.9 and 83.4%, respectively.

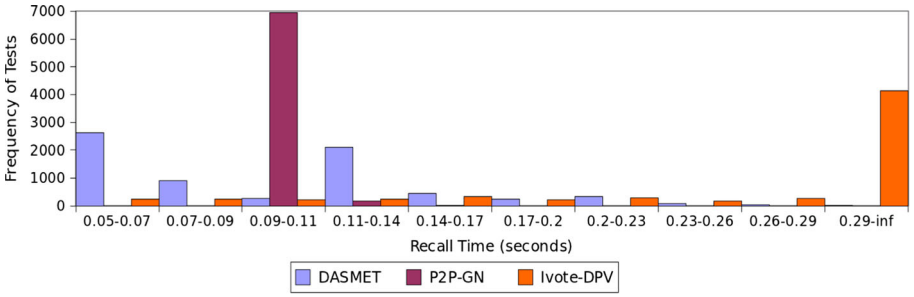
In the other hand, in a group of tests which required less than 0.09 s recall time, about 55, 45 and 56.4% of the DASMET tests required less than 0.09 s recall time in the FCTH, CEDD and Haralick experiments, respectively. The group also comprises around 6.9% of the Ivote-DPV tests in the FCTH experiment, 7% of the Ivote tests in the CEDD experiment and 6.4% of the Ivote-DPV tests in the Haralick experiment. In the Haralick experiment, 90.6% of the P2P-GN tests were completed within 0.09 s; however, none of them took less than 0.09 s in the FCTH and CEDD experiments.

Although the P2P-GN demonstrated a fast recall time in the Haralick experiment, the high proportion of the DASMET tests that used low recall time (<0.09 s) in all of the three experiments, plus the high accuracy of results, demonstrates the ability of the DASMET to provide an accurate prediction with a quick response time. Furthermore, Histogram 2 in Fig. 12b shows that about 42.4% of the DASMET tests and none of the P2P-GN tests are completed in less than 0.07 s in the Haralick experiment. This suggests that the DASMET has improved the recall speed of at least 42.4% tests of the P2P-GN, although applying the DASMET may delay about 11% of the tests as a trade-off. This further emphasises the effectiveness and efficiency of the DASMET.

The delay of recall time ($\hbar$ times the recall time) , particularly in the worst case scenario, was caused by $\hbar$ factor. However, the impact of $\hbar$ can be significantly reduced when the communication delay per hop is small.



(a) Histogram 1 for recall time per test on the CEDD feature set



(b) Histogram 2 for recall time per test on the CEDD feature set

Fig. 11 Recall time per test in the CEDD experiment

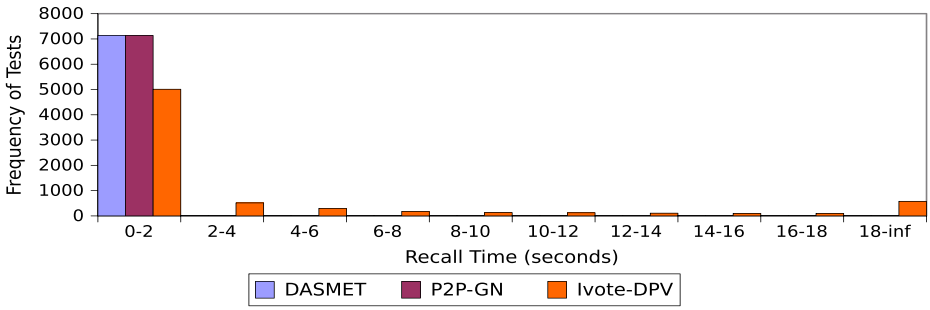
7 Complexity analysis

In this section, we analyse the overall complexity at the network level which covers the use of resources in the whole learning or recall operation (taking into account the computational overhead at all participating peers)—starting from the submission of queries at a requester to the end of the learning or recall process. Let d be the data dimension (number of attributes), n_H be the number of sub-patterns or leaf nodes, H_l be a set of nodes at level-l and $\hbar$ be the height of the DASMET tree. The magnitude of the sequential steps is defined by the height of the tree, that is $\hbar$ which becomes a factor that influences the run-time and communication overheads.

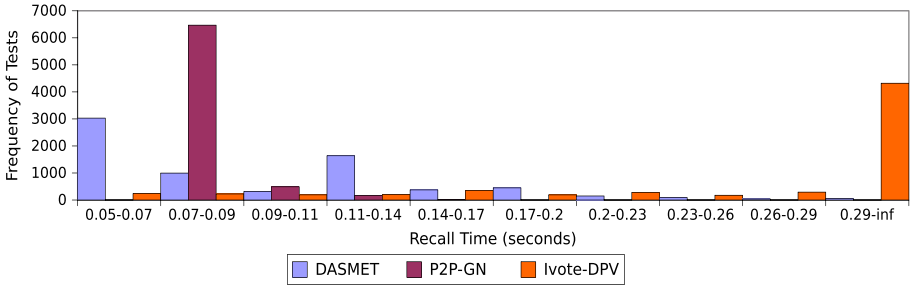
Nonetheless, $\hbar$ is a small positive integer and $\hbar$ is $\mathcal{O}(\log\ n_H)$ and n_H is $\mathcal{O}(d)$. Since n_H is $\mathcal{O}(d)$, then $\hbar = \mathcal{O}(\log\ d)$. This shows that the height of the tree increases slowly with an increase in data dimension.

Let n_R be the total number of nodes within the DASMET tree (see Equation (2)); $M[DASMET]_{learn}$ be the number of messages incurred per learning process; and $M[DASMET]_{recall}$ be the number of messages incurred per recall process. For the implementation on a network with optimal connections,[7] $M[DASMET]_{learn}$ and $M[DASMET]_{recall}$ are $\Theta(n_R)$ messages while, in the implementation within a Chord P2P network, they are $\mathcal{O}(n_R \cdot \log N)$ messages. In the best-case scenario, $M[DASMET]_{recall}$

[7] Every peer knows the location of all the others, so that direct connections among them can be established.



(a) Histogram 1 for recall time per test on the Haralick feature set



(b) Histogram 2 for recall time per test on the Haralick feature set

Fig. 12 Recall time per test in the Haralick experiment

equals two messages since a valid prediction is obtained at the first phase and the lookup function only requires one hop message.

The overall learning time per instance, $t_O[L_{DASMET}]$ and the overall recall time per instance, $t_O[R_{DASMET}]$. $t_O[L_{DASMET}]$ is given in (3).

$$t_O[L_{DASMET}] = t_{LQ} + t_{localLearn} \tag{3}$$

where $t_{localLearn}$ is the time for local learning processes at leaf nodes and t_{LQ} is the communication time to send `LEARN_REQUEST` messages.

In the worst case scenario, the recall procedure involves all $(\hbar + 1)$ recall phases where the first phase involves nodes at level-0, the second phase involves nodes at level-1 and the last phase involves nodes at level-$\hbar$. In the lth recall phase, $|H_l|$ `RECALL_REQUEST` messages are sent in parallel to H_l (all DASMET nodes at level l). Therefore, the upper bound of $t_O[R_{DASMET}]$ is given in (4) as below.

$$\begin{aligned} t_O[R_{DASMET}] = \sum_{l=0}^{\hbar-1} \big(t_{RQ}[l] + t_{localRecall}[l] + t_{RR}[l] + t_{agg}[l]\big) + \\ t_{RQ}[\hbar] + t_{localRecall}[\hbar] + t_{RR}[\hbar] + t_{agg}[\hbar] + t_{fP} \end{aligned} \tag{4}$$

where $t_{localRecall}[l]$ is the execution time of the function localRecall(.) at nodes at level-l; $t_{agg}[l]$ is the execution time of the function agg(.) at a requester to aggregate predictions from all nodes at level-l; $t_{RQ}[l]$ is the communication time to send `RECALL_REQUEST` messages for all nodes at level-l; $t_{RR}[l]$ is the communication time to send `RECALL_RESPONSE` messages from all nodes at level-l; and t_{fP} is the execution time

Table 5 Summary of the DASMET complexity

	Overall computational complexity	
	Chord network	Optimal connections
$M[DASMET]_{learn}$	$\mathcal{O}(n_R \cdot \log N)$	$\Theta(n_R)$
$M[DASMET]_{recall}$	$\mathcal{O}(n_R \cdot \log N)$	$\Theta(n_R)$
$t_O[L_{DASMET}]$	$\mathcal{O}(\log\ N)$	$\mathcal{O}(1)$
$t_O[R_{DASMET}]$	$\mathcal{O}((d + \log\ N) \log d)$	$\mathcal{O}(d \log d)$

of the function finalPrediction(.) at a requester. Note that the function finalPrediction(.) is only executed at the last step of the recall procedure.

Then, the best-case recall performance is given in (5) as below.

$$t_O[R_{DASMET}]_{best} = t_{RQ}[0] + t_{localRecall}[0] + t_{RR}[0] + t_{agg}[0] + t_{fP} \tag{5}$$

To simplify these calculation, we assume that t_{RR} is equal to 1, and t_{RQ} and t_{LQ} are denoted by n_{hops} since these involve a lookup process. $t_{localRecall}$, $t_{localLearn}$ and t_{fP} are denoted by t_1. Then, t_{agg} is denoted by $\tilde{s} \times t_1$ for $\tilde{s}$ inputs. By applying these assumptions into (3) and (4), the estimated $t_O[L_{DASMET}]$ and $t_O[R_{DASMET}]$ are given in (6) and (7), respectively.

$$\begin{aligned} t_O[L_{DASMET}] &= n_{hops} + t_1 \\ &= \mathcal{O}(n_{hops}) \end{aligned} \tag{6}$$

$$\begin{aligned} t_O[R_{DASMET}] &= \hbar[n_{hops} + \tilde{s} \cdot t_1 + t_1 + 1] + [n_{hops} + \tilde{s} \cdot t_1 + 2 \cdot t_1 + 1] \\ &= (\hbar + 1)[n_{hops} + \tilde{s} \cdot t_1 + t_1 + 1] + t_1 \end{aligned} \tag{7}$$

By substituting $\hbar = \mathcal{O}(\log\ d)$ and $\tilde{s} = \mathcal{O}(d)$ in (7), $t_O[R_{DASMET}]$ is $\mathcal{O}((d+n_{hops}) \log\ d)$. The aggregation time for level-0 $t_{agg[0]}$ equals t_1 since $\tilde{s} = 1$. Accordingly, by substituting this into (5), the estimated $t_O[R_{DASMET}]_{best}$ is given in (8).

$$\begin{aligned} t_O[R_{DASMET}]_{best} &= n_{hops} + 3 \cdot t_1 + 1 \\ &= \mathcal{O}(n_{hops}) \end{aligned} \tag{8}$$

The estimated communication overhead and recall time in an implementation with optimal connections can simply be obtained by replacing n_{hops} value to 1 in (6) and (7), while n_{hops} equals $\mathcal{O}(\log N)$ in an implementation within a Chord network. This is summarised in Table 5.

This analysis shows that the DASMET preserves the efficiency and scalability of the P2P-GN. Nonetheless, the advantage of the DASMET is that its best-case performance incurs a significantly low communication overhead and recall run-time, particularly for a duplicate of a stored pattern—with optimal connections, the best-case recall performance incurs a constant recall time and two messages. Hence, it encourages a fast recall time in the environment with a large number of duplicate and near-duplicate patterns.

8 Conclusion and future works

In this paper, we have presented an efficient algorithm for a fully-distributed near-duplicates detection—the DASMET. The algorithm is able to detect exact, near-duplicates and fuzzy patterns in a small number of iterations (at most $\hbar$ iterations). We have reported the application of the DASMET in the image spam detection.

The DASMET generally has the best accuracy compared to centralised, state-of-the-art classifiers (naive Bayes, 1-NN, *k*-NN, RBFN and BPNN) and other distributed classifiers (P2P-GN, ensemble *k*-NN, ensemble naive Bayes and Ivote-DPV) in the spam detection experiments with 97% to 99% accuracy in all the experiments, using different feature sets (Haralick descriptor, CEDD descriptor and FCTH descriptor). Its false positive rate is less than 0.02 in all spam detection experiments and this implies that it rarely misclassified legitimate images as spam.

We found that the Haralick descriptor gives a better prediction than the CEDD and FCTH descriptor for the image spam problem in this study. We also found that the recall time and the communication overhead during recall in the P2P-GN are uniformly distributed, while those in the Ivote-DPV and the DASMET are spuriously distributed.

In the worst case scenario of recall process, the worst communication overhead (highest number of messages per test) in the Ivote-DPV is higher than the worst communication overhead in the DASMET. The worst case communication overhead in the DASMET is, however, higher than the P2P-GN. Nonetheless, the communication overheads in the DASMET is positively skewed where approximately 87% of the DASMET recall tests used less than 100 messages in the CEDD and FCTH experiments, while none of the P2P-GN recall tests used than 100 messages in those experiments.

Our results in the FCTH and CEDD experiments have shown that none of the P2P-GN recall tests was completed in less than 0.09 seconds. However, 55% of the DASMET recall tests in the FCTH experiment and 45% of the DASMET recall tests in the CEDD experiment were completed in less than 0.09 seconds. In the Haralick experiment, the DASMET improves the recall time of at least 42.4% of the P2P-GN tests, but delay about 11% of the tests. These results show the efficiency of the DASMET to provide faster responses in most of the tests compared to the P2P-GN, but with the trade-off of slower responses in a small percentage of the tests.

The worst case recall time and communication overheads of the DASMET are influenced by $\hbar$ factor which is equal to the height of the DASMET tree. Our theoretical analysis shows that $\hbar$ increases slowly with d. Hence, the number of iterations is significantly small in comparison to NN, boosting, DMV-based and DPV-based algorithms.

In conclusion, DASMET reduces the classification overheads for the type of problem with a high number of duplicate and near-duplicate data. The DASMET algorithm is introduced to improve the communication overheads and recall time in the P2P-GN by detecting the duplicates or closely resembling patterns at the entry point of the system. Hence, it avoids wasting resources for expensive pattern recognition processes for this kind of data. This is accomplished without losing the ability to recognize fuzzy data.

We have experimentally demonstrated that the DASMET improves the communication overhead in the P2P-GN. Approximately 87% of the DASMET recall tests used less than 100 messages in the CEDD and FCTH experiments, while none of the P2P-GN recall tests used the same number of messages. Compared to the Ivote-DPV, the DASMET is more communication-efficient and time-efficient. All of the P2P-GN recall tests and the DASMET recall tests were completed in less than 0.29 seconds, while only approximately 36.4% of the Ivote-DPV recall tests were completed in less than 0.29 seconds.

The effectiveness of DASMET algorithms in the evaluation on the image spam problem shows the potential of our solution in dealing with spam problems. In the image spam experiments, it generally has a better accuracy than other methods (NN, *k*-NN, naive Bayes, BPNN and RBFN, Ivote-DPV, ensemble *k*-NN, ensemble naive Bayes and P2P-GN).

Since the paper focusses on the distributed classification part instead of feature extraction, we randomly selected three descriptors (Haralick, CEDD, and FCTH descriptors) to

evaluate our algorithm for this problem. However, we plan to test DASMET by using other low-level features such as local binary pattern (LBP), Tamura and other feature descriptors to study their performance for the problem in the future. Due to their high complexity, Support Vector Machine (SVM) and Region-based Convolutional Neural Networks (R-CNN) [16] were not evaluated in this paper, but we intend to compare our algorithm with SVM and R-CNN in our future works by using higher performance machine. Nonetheless, we have achieved our objective to show that our algorithm is comparable to the state-of-the-art algorithms.

Acknowledgement The research reported in this paper is supported by Research Acculturation Grant Scheme (RAGS) 9018-00080. The authors would also like to express gratitude to the Malaysian Ministry of Higher Education (MOHE) and University Malaysia Perlis (UniMAP) for the facilities provided.

Appendix A: Other Algorithms

A.1 Tree construction

Algorithm 3 are executed to generate a logical DASMET tree. The process of constructing the logical tree is recursive and it starts from a root node. Let level be 0, $\widehat{X} = \{\hat{x}_i\}_{i=1}^{n_h}$ be a set of sub-patterns, w be the number of sub-patterns (n_H), m be the maximum number of children of each node and d_s be the segment size at a leaf node. Note that m in this algorithm is equal to φ_s. All segments in $\widehat{X}$ are initially assigned to the root node V where the following steps in function constructTree(level,w, $\widehat{X}$, m, H) as explained in Algorithm 3 are executed.

Algorithm 3 constructTree(level,w, $\widehat{X}$, m)

```
level ← level+1
if w ≤ m then
    create w leaf nodes H = {h_1, h_2, ··· h_w}
    assign x̂_i ∈ X̂ to leaf node h_i ∈ H
else
    n_c ← 0
    if ⌊w/m⌋ < m then
        n_c ← ⌊w/m⌋ + 1
    else
        n_c ← m
    end if
    n ← ⌊w/n_c⌋
    balance ← w − (n × n_c)
    y ← 0
    while y < n_c do
        w ← n
        if balance > 0 then
            w ← w + 1
            balance ← balance − 1
        end if
        create a child node h_i
        assign w segments, X̂ = {x̂_1, x̂_2, ··· x̂_w} to h_i
        constructTree(level,w, X̂, m)
        y ← y + 1
    end while
end if
```

The node firstly determines whether it should expand the tree or not. In case that w is less or equal to m, then the node creates w leaf nodes and assigns one segment per leaf node; this completes the process. Otherwise, it determines the number of children n_c using (9) as below.

$$n_c = \begin{cases} m & \text{if } \lfloor \frac{w}{m} \rfloor \geq m \\ \lfloor \frac{w}{m} \rfloor + 1 & \text{otherwise} \end{cases} \tag{9}$$

Next, it creates n_c child nodes and distributes the available segments to these child nodes using greedy approach. Upon receiving w segments from its parent, every child node then executes Algorithm 3. This process is executed recursively until $w \leq m$.

A.2 Generate Identifier

Algorithm 4 generatingIdentifier($\vec{x}$)

1: $l \leftarrow \hbar$
2: Create segments to learn, $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_{n_H}\}$ from $\vec{x}$
3: $i \leftarrow 1$
4: **while** $i \leq n_H$ **do**
5: Calculate bias identifier for a leaf node h_i, $\psi(h_i) \leftarrow \mathsf{qLeaf}(\hat{x}_i, i)$.
6: $i \leftarrow i + 1$
7: **end while**
8: $l \leftarrow l - 1$
9: **while** $l \geq 0$ **do**
10: $i \leftarrow 1$
11: $m_l \leftarrow$ number of nodes at level-l.
12: **while** $i \leq m_l$ **do**
13: $\{\psi_{(j,i)}\}_{j=1}^{w_i}$ is a set of bias identifiers of w_i child nodes of internal/root node i.
14: Calculate bias identifier for internal/root node h_i, $\psi(h_i) \leftarrow \mathsf{q}(\psi_{(j,i)}\}_{j=1}^{w_i}$
15: $i \leftarrow i + 1$
16: **end while**
17: $l \leftarrow l - 1$
18: **end while**

References

1. Amir A, Srinivasan B, Khan A (2015) A communication-efficient distributed algorithm for large-scale classification within P2P networks. In: Proceedings of the 6th international symposium on information and communication technology, SoICT 2015. ACM, NY, USA. ISBN 978-1-4503-3843-1, pp 75–82. doi:10.1145/2833258.2833304
2. Amir A, Amin AHM, Khan A (2013) Developing machine intelligence within p2p networks using a distributed associative memory. In: Dowe DL (ed) Algorithmic probability and friends. Bayesian prediction and artificial intelligence: papers from the Ray Solomonoff 85th memorial conference, Melbourne, VIC, Australia, 2011. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-44958-1, pp 439–443. doi:10.1007/978-3-642-44958-1_35
3. Attar A, Rad RM, Atani RE (2013) A survey of image spamming and filtering techniques. Artif Intell Rev 40(1):71–105. ISSN 1573–7462. doi:10.1007/s10462-011-9280-4
4. Alazab M, Broadhurst R (2015) The role of spam in cybercrime: data from the Australian cybercrime pilot observatory. In: Smith RG, Cheung RC-C, Lau LY-C (eds) Cybercrime risks and responses: eastern and western perspectives, Palgrave Macmillan UK, London, ISBN 978-1-137-47416-2, pp 103–120. doi:10.1057/9781137474162_7

5. Al-Duwairi B, Khater I, Al-Jarrah O (2012) Detecting image spam using image texture features. Int J Inf Secur Res (IJISR) 2(3/4):344–353
6. Blanzieri E, Bryl A (2008) A survey of learning-based techniques of email spam filtering. Artif Intell Rev 29(1):63–92
7. Bouckaert RR, Frank E, Hall M, Kirkby R, Reutemann P, Seewald A, Scuse D (2014) WEKA Manual for Version 3-7-11, http://www.cs.waikato.ac.nz/ml/weka/documentation.html
8. Chatzichristofis SA, Boutalis YS (2008) CEDD: Color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In: Gasteratos A, Vincze M, Tsotsos J (eds) Computer vision systems, vol 5008 of lecture notes in computer science. Springer Berlin Heidelberg, pp 312–322
9. Chatzichristofis SA, Boutalis YS (2008) FCTH: Fuzzy color and texture histogram - a low level feature for accurate image retrieval. In: Proceedings of the 2008 9th international workshop on image analysis for multimedia interactive services, (WIAMIS '08), Klagenfurt, Austria, WIAMIS '08, IEEE Computer Society. Washington, DC, USA, pp 191–196
10. Chen J, Zhao H, Yang J, Zhang J, Li T, Wang K (2015) An intelligent character recognition method to filter spam images on cloud. Soft Computing. 1-11ISSN 1433-7479. doi:10.1007/s00500-015-1811-5
11. Chowdhury M, Gao J, Chowdhury M (2015) Image spam classification using neural network. In: Thuraisingham B, Wang X, Yegneswaran V (eds) Security and privacy in communication networks: 11th international conference, SecureComm 2015, Dallas, TX, USA, October 26-29, 2015, Revised Selected Papers, Springer International Publishing, Cham, ISBN 978-3-319-28865-9, pp 622–632. doi:10.1007/978-3-319-28865-9_41
12. Dredze M, Gevaryahu R, Elias-Bachrach A (2007) Learning fast classifiers for image spam. In: Fourth conference on email and anti-spam, (CEAS 2007). Mountain View, California
13. Filasiak R, Grzenda M, Luckner M, Zawistowski P (2014) On the testing of network cyber threat detection methods on spam example. Ann Telecommun - Annal Télécommun 69(7):363–377. ISSN 1958-9395. doi:10.1007/s12243-013-0412-5
14. Gao Y, Yang M, Zhao X, Pardo B, Wu Y, Pappas TN, Choudhary A (2008) Image spam hunter. In: 2008 IEEE international conference on acoustics, speech and signal processing, ISSN 1520-6149, pp 1765–1768. doi:10.1109/ICASSP.2008.4517972
15. Geusebroek J-M, Burghouts GJ, Smeulders AWM (2005) The Amsterdam library of object images. Int J Comput Vis 61(1):103–122
16. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE conference on computer vision and pattern recognition, ISSN 1063-6919, pp 580–587. doi:10.1109/CVPR.2014.81
17. Gupta R, Singha N, Singh YN (2015) Reputation based probabilistic resource allocation for avoiding free riding and formation of common interest groups in unstructured P2P networks. Peer-to-Peer Networking and Applications, pp 1–13
18. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor Newslett 11(1):10–18
19. Haralick R, Shanmugam K, Dinstein I (1973) Textural features for image classification. IEEE Trans Syst Man Cybern 3(6):610–621
20. Jin X, Chan S-HG (2010) Detecting malicious nodes in peer-to-peer streaming by peer-based monitoring. ACM Trans Multimed Comput Commun Appl 6(2):9:1–9:18
21. JFeatureLib, JFeatureLib: A free java library containing feature descriptors and detectors, [Online viewed on April 6, 2013] http://code.google.com/p/jfeaturelib/, 2013
22. Kapelko R (2013) Towards fault-tolerant chord p2p system: analysis of some replication strategies. In: Ishikawa Y, Li J, Wang W, Zhang R, Zhang W (eds) Web technologies and applications, vol 7808 of lecture notes in computer science. Springer, Berlin Heidelberg, pp 686–696
23. Kurdi HA (2015) HonestPeer: An enhanced EigenTrust algorithm for reputation management in fP2Pg systems. J King Saud Univ - Comput Inf Sci 27(3):315–322
24. Luo P, Xiong H, Lü K, Shi Z (2007) Distributed classification in Peer-to-Peer networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, (KDD '07). San Jose, California, USA, pp 968–976
25. Maldonado S, L'Huillier G (2013) SVM-based feature selection and classification for email filtering. In: Latorre Carmona P, Snchez JS, Fred AL (eds) Pattern recognition - applications and methods, vol 204 of advances in intelligent systems and computing. Springer Berlin Heidelberg, pp 135–148

26. Mehta B, Nangia S, Gupta M, Nejdl W (2008) Detecting image spam using visual features and near duplicate detection. In: Proceedings of the 17th international conference on world wide web, (WWW'08). Beijing, China, pp 497–506
27. Montresor A, Jelasity M (2009) PeerSim: A scalable P2P simulator. In: Proceedings of the 9th international conference on peer-to-peer, (P2P'09). Seattle, Washington, USA, pp 99–100
28. Ozgur L, Gungor T, Gurgen F (2004) Spam mail detection using artificial neural network and bayesian filter. In: Yang Z, Yin H, Everson R (eds) Intelligent data engineering and automated learning IDEAL 2004, vol 3177 of lecture notes in computer science. Springer Berlin Heidelberg, pp 505–510
29. Ruan G, Tan Y (2010) A three-layer back-propagation neural network for spam detection using artificial immune concentration. Soft Comput 14(2):139–150. doi:10.1007/s00500-009-0440-2
30. Sig2Dat, Sig2Dat Website, [Online viewed on April 6, 2016] http://sourceforge.net/projects/sig2dat/, 2016
31. Vieira AB, De Almeida RB, De Almeida JM, Campos SVA (2013) SimplyRep: A simple and effective reputation system to fight pollution in fP2Pg live streaming. Comput Netw 57(4):1019–1036. ISSN 1389-1286
32. Wakade S, Liszka KJ, Chan C-C (2013) Application of learning algorithms to image spam evolution. In: Ramanna S, Jain CL, Howlett JR (eds) Emerging paradigms in machine learning, Springer Berlin Heidelberg. Berlin, Heidelberg, ISBN 978-3-642-28699-5, pp 471–495, doi:10.1007/978-3-642-28699-5_18
33. Zhang C, Huang L (2015) Study on content-based of image retrieval. In: Zhang R, Zhang Z, Liu K, Zhang J (eds) LISS 2013: Proceedings of 3rd international conference on logistics, informatics and service science. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-40660-7, pp 591–594, doi:10.1007/978-3-642-40660-7_87
34. Zhou F, Zhuang L, Zhao BY, Huang L, Joseph AD, Kubiatowicz J (2003) Approximate object location and spam filtering on peer-to-peer systems. In: Proceedings of the ACM/IFIP/USENIX 2003 international conference on middleware, (Middleware '03). Rio de Janeiro, Brazil, pp 1–20
35. Zuo M, Ma Y-H, Chbeir R, Li J-H (2007) Combating P2P file pollution with co-alerting. In: Proceedings of the 2007 3rd international IEEE conference on signal-image technologies and internet-based system (SITIS 2007). Shanghai, China, pp 289–297

Amiza Amir is a senior lecturer in School of Computer and Communication Engineering at Universiti Malaysia Perlis. She received her Ph.D. in Information Technology, on distributed artificial intelligence, from Monash University, Australia in 2015. Her current research interests include distributed data mining, cloud computing, optimization system, and software-defined network (SDN). She teaches courses in C programming and Object-oriented programming.

Bala Srinivasan is a professor of Information Technology in the Faculty of Information Technology, Monash University, Australia. He was formerly an academic staff member of the Department of Computer Science and Information Systems at the National University of Singapore, Singapore and the Indian Institute of Technology, Kanpur, India. He has more than 30 years of experience in academia, industries and research organizations. He has authored and jointly edited 6 technical books and more than 350 refereed publications in international journals and conferences in the areas of multimedia databases, data communications, data mining and distributed systems, and has attracted a number of research grants. He has successfully supervised more than 40 PhDs and 15 Masters by research theses. He is a founding chairman of the Australasian Database Conference. He is a gold medalist in Bachelor of Engineering Honours degree in Electronics and Communication Engineering, Guindy Engineering College, University of Madras, a Masters and a PhD degree both in Computer Science from the Indian Institute of Technology, Kanpur, India.

Asad Khan works in the Faculty of Information Technology at Monash University as a senior lecturer. He was awarded a Ph.D. in 1994 by Faculty of Engineering at Heriot-Watt. His work on parallel domain decomposition using natural programming techniques led to large research grants from British Science & Research Council and leading industrial bodies. He was appointed a lecturer at Heriot-Watt in 1993 and later took up a computer centre management role at Monash University in Australia. During this period he was involved with the design of large storage and high performance computing projects. He was appointed a senior lecturer in Faculty of Information Technology at Monash University in 2000. He is a recipient of several large grants from Australian Research Council and Department of Education Science & Training. His applied research involves development of E-Research systems and intelligent sensor networks. His theoretical research areas include natural computation, neural networks, and pattern recognition.