Finding Hidden Group Structure in a Stream of Communications*

J. Baumes[1], M. Goldberg[1], M. Hayvanovych[1], M. Magdon-Ismail[1], W. Wallace[2], and M. Zaki[1]

[1] CS Department, RPI, Rm 207 Lally, 110 8th Street, Troy, NY 12180, USA.
Email: {baumej,goldberg,hayvam,magdon,zaki}@cs.rpi.edu.
[2] DSES Department, RPI, 110 8th Street, Troy, NY 12180, USA.
Email: wallaw@rpi.edu.

Abstract. A *hidden group* in a communication network is a group of individuals planning an activity over a communication medium without announcing their intentions. We develop algorithms for separating non-random planning-related communications from random background communications in a streaming model. This work extends previous results related to the identification of hidden groups in the cyclic model. The new statistical model and new algorithms do not assume the existence of a planning time-cycle in the stream of communications of a hidden group. The algorithms construct larger hidden groups by building them up from smaller ones. To illustrate our algorithms, we apply them to the Enron email corpus in order to extract the evolution of Enron's organizational structure.

1 Introduction

Modern communication networks (telephone, email, Internet chat room, etc.) facilitate rapid information exchange among millions of users around the world. This vast communication activity provides the ideal environment for groups to plan their activity undetected: the related communications are embedded (hidden) within the myriad of random background communications, making them difficult to discover. When a number of individuals in a network exchange communications related to a common goal, or a common activity, they form a group; usually, the presence of the coherent communication activity imposes a certain structure on the communications of that set (group) of actors. A group of actors may communicate in a structured way while not being forthright in exposing its existence and membership. In this paper, we describe a novel statistical and algorithmic approach to discovering such hidden groups. Our work extends previous results related to the identification of hidden groups in the cyclic model.

* This material is based upon work partially supported by the National Science Foundation under Grant No. 0324947. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The new statistical model and new algorithms do not assume the existence of a planning time-cycle in the stream of communications of a hidden group.

The tragic events of September 11, 2001 underline the need for algorithmic tools (and corresponding software implementations) which facilitate the discovery of hidden (malicious) groups during their *planning* stage, before they move to implement their plans. A generic way of discovering such groups is based on discovering *correlations* among the communications of the actors in the communication network. Although the content of the messages can be informative and natural language processing may be brought to bear in its analysis, such an analysis is generally time consuming and intractable for large datasets. The research presented here makes use of only three properties of a message: its time, the name of the sender and the name of the recipient of the message.

Our approach is based on the observation that a pattern of communications exhibited by actors in a social group pursuing a common objective is different from that of a randomly selected set of actors. Specifically, we focus on the discovery of such groups whose communications during the observation time-period exhibit statistical *correlations*. Since any group, even one which tries to hide itself, must communicate regularly, hidden groups will have communications that display statistically significant structure, as compared to a group formed at random. This is related to the social concept of homophily, which states that individuals will tend to communicate with those similar to themselves [1].

The study of identifying hidden groups was initiated in [2] using Hidden Markov models. In [3, 4], algorithms were established for detecting groups which are correlated in time, given certain assumptions. In particular, it was assumed that the group communicates among all members at least once over consecutive disjoint time intervals of a given length – the *cycle* model. The contribution of this paper is the formulation of the problem of finding hidden groups in a streaming model (*streaming* hidden groups), together with algorithms for finding such hidden groups. In this model, hidden groups do not necessarily display a fixed time-cycle, during which all members of group members exchange messages. An example of a streaming hidden group is illustrated in Figure 1(a) – a group planning a golf game. Given the message content, it is easy to identify two "waves" of communication. The first wave (in darker font) establishes the golf game; and, the second wave (in lighter font) finalizes the game details. Based on this data, it is not hard to identify the group and conclude that the "organizational structure" of the group is represented in Figure 2 to the right (each actor is represented by their first initial). The challenge is to deduce this same information from the communication stream *without* the message contents (Figure 1(b)). There are two main features that distinguish the stream model from the cycle model:

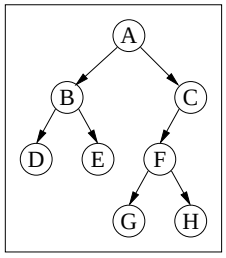


Fig. 2. Group structure for Figure 1

(i) communication waves may overlap, as in Figure 1(a);

(ii) waves may different durations, some considerably longer than others.

00 **A→C** Golf tomorrow? Tell everyone.	00 **A→C**
05 **C→F** Alice mentioned golf tomorrow.	05 **C→F**
06 **A→B** Hey, golf tomorrow? Spread the word	06 **A→B**
12 **A→B** Tee time: 8am; Place: Pinehurst.	12 **A→B**
13 **F→G** Hey guys, golf tomorrow .	13 **F→G**
13 **F→H** Hey guys, golf tomorrow .	13 **F→H**
15 **A→C** Tee time: 8am; Place: Pinehurst.	15 **A→C**
20 **B→D** We're playing golf tomorrow.	20 **B→D**
20 **B→E** We're playing golf tomorrow.	20 **B→E**
22 **C→F** Tee time: 8am; Place: Pinehurst.	22 **C→F**
25 **B→D** Tee time: 8am; Place: Pinehurst.	25 **B→D**
25 **B→E** Tee time 8am, Pinehurst.	25 **B→E**
31 **F→G** Tee time 8am, Pinehurst.	31 **F→G**
31 **F→H** Tee off 8am,Pinehurst.	31 **F→H**
(a)	(b)

Fig. 1. (a) Streaming hidden group with two waves of planning. (b) Streaming group without message content – only time, sender id and receiver id are available.

The first feature may result in bursty waves of intense communication (many overlapping waves) followed by periods of silence. Such a type of communication dynamics is hard to detect in the cycle model, since all the (overlapping) waves of communication may fall in one cycle. The second can be quantified by a propagation delay function which specifies how much time may elapse between a hidden group member receiving the message and forwarding it to the next member; sometimes the propagation delays may be large, and sometimes small. One would typically expect that such a streaming model would be appropriate for hidden groups with some organizational structure as illustrated in the tree. We present algorithms which not only discover the streaming hidden group, but also its organizational structure *without the use of message content.*

We use the notion of communication frequency in order to distinguish non-random behavior. Thus, if a group of actors communicate unusually often using the same chain of communication, i.e. the structure of their communications persists through time, then we consider this group to be statistically significant and indicative of a hidden group. We present algorithms to detect small frequent tree-like structures, and build larger hidden structures starting from the small ones.

Paper Organization. We begin in Section 2 by formally describing the problem and the data representation. We present our algorithms and statistical models for discovering streaming hidden groups in Sections 3, **??** and 5. In Section 6, we give some experimental results on the evolution of the Enron organizational structure using the Enron e-mail corpus, and conclude in Section 7 with some future directions.

2 Problem Statement

A communication stream is a set of tuples of the form $\langle \mathsf{senderID}, \mathsf{receiverID}, \mathsf{t}, \mathsf{msg} \rangle$, where $\mathsf{senderID}$ sends the message msg to $\mathsf{receiverID}$ at time t. In our approach to detecting hidden groups, we do not rely on any semantic information (message content) contained in the communications. The reason is that communications on a public network are usually encrypted in some way and can be quite complex to analyze, hence the message information may be either misleading or unavailable.

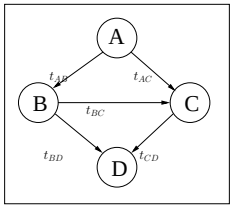


Fig. 3. Hypothetical Group Structure

A hidden group communication structure can be represented by a directed graph. Each vertex is an actor and every edge shows the direction of the communication. For example a hierarchical organization structure could be represented by a directed tree. The graph in Figure 3 to the right is an example of a communication structure, in which actor A "simultaneously" sends messages to B and C; then, after receiving the message from A, B sends messages to C and D; C sends a message to D after receiving the messages from A and B. Every graph has two basic types of communication structures: *chains* and *siblings*. A *chain* is a path of length at least 3, and a *sibling* is a tree with a root and two or more children, but no other nodes. Of particular interest are chains and sibling trees with three nodes, which we denote *triples*. For example, the chains and sibling trees of size three (triples) in the communication structure above are: $A \rightarrow B \rightarrow D$; $A \rightarrow B \rightarrow C$; $A \rightarrow C \rightarrow D$; $B \rightarrow C \rightarrow D$; $A \rightarrow B, C$; and, $B \rightarrow C, D$. We suppose that a hidden group employs a communication structure that can be represented by a directed graph as above. If the hidden group is hierarchical, the communication graph will be a tree. The task is to discover such a group and its structure based solely on the communication data.

If a communication structure appears in the data many times, then it is likely to be non-random, and hence represent a hidden group. To discover hidden groups, we will discover the communication structures that appear many times. We thus need to define what it means for a communication structure to "appear". Specifically, we consider chain and sibling triples (trees of size three). For a chain $A \rightarrow B \rightarrow C$ to appear, there must be communication $A \rightarrow B$ at time t_{AB} and a communication $B \rightarrow C$ at time t_{BC} such that $(t_{BC} - t_{AB}) \in [\tau_{min}, \tau_{max}]$. This intuitively represents the notion of causality, where $A \rightarrow B$ "causes" $B \rightarrow C$ within some time interval specified by τ_{min}, τ_{max}. A similar requirement holds for the sibling triple $A \rightarrow B, C$; the sibling triple appears if there exists t_{AB} and t_{AC} such that $(t_{AB} - t_{AC}) \in [-\delta\ \delta]$. This constraint represents the notion of B sending messages "simultaneously" to C and D which are within a small time of each other, as specified by δ. For an entire graph (such as the one above) to appear, every chain and sibling triple in the graph must appear using a single set of times. For example, in the graph example above, there must exist a set of times, $\{t_{AB}, t_{AC}, t_{BC}, t_{BD}, t_{CD}\}$, which satisfies all the six chain and sibling

constraints. A graph appears multiple times if there are disjoint sets of times each of which is an appearance of the graph. A set of times *satisfies* a graph if all chain and sibling constraints are satisfied by the set of times. The number of times a graph appears is the maximum number of disjoint sets of times that can be found, where each set satisfies the graph. Intuitively, causality requires that multiple occurrences of a graph should monotonically increase in time. Specifically, if t_{AB} "causes" t_{BC} and t'_{AB} "causes" t'_{BC} with $t'_{AB} > t_{AB}$, then it should be that $t'_{BC} > t_{BC}$. In general, if we have two disjoint occurrences (sets of times) $\{t_1, t_2, \ldots\}$ and $\{s_1, s_2, \ldots\}$ with $s_1 > t_1$, then it should be that $s_i > t_i$ for all i.

A communication structure which occurs frequently enough becomes statistically significant when its frequency of occurrence exceeds the expected frequency of such a structure from the random background communications. The goal is to find all statistically significant communication structures, which is formally stated in the following algorithmic problem statement.

Input: A communication data stream; δ, τ_{min}, τ_{max}, h, κ.
Output: All communication structures of size $\geq h$, which appear at least κ times, where the appearance is defined with respect to δ, τ_{min}, τ_{max}.

The statistical task is to determine h and κ to ensure that all output communication structures are statistically significant. We will first consider small trees, specifically chain and sibling triples. We then develop a heuristic algorithm to build up larger hidden groups from clusters of triples. We will also obtain evolving hidden groups by using a sliding window.

3 Algorithms for Chain and Sibling Trees

We will start by introducing a technique to find chain and sibling triples, i.e. trees of type $A \rightarrow B \rightarrow C$ (chain) and trees of type $A \rightarrow B, C$ (sibling). To accomplish this, we will enumerate all the triples and count the number of times each triple occurs. Enumeration can be done by brute force, i.e. considering each possible triple in the stream of communications. We have developed a general algorithm for counting the number of occurrences of chains of length ℓ, and siblings of width k. These algorithms proceed by posing the problem as a multi-dimensional matching problem, which in the case of tipples becomes a two-dimensional matching problem. Generally multi-dimensional matching is hard to solve, but in our case the causality constraint imposes an ordering on the matching which allows us to construct a linear time algorithm. Finally we will introduce a heuristic to build larger graphs from statistically significant triples using overlapping clustering techniques [5].

3.1 Computing the Frequency of a Triple

Consider the triple $A \rightarrow B \rightarrow C$ and the associated time lists $L_1 = \{t_1 \leq t_2 \leq \ldots \leq t_n\}$ and $L_2 = \{s_1 \leq s_2 \leq \cdots \leq s_m\}$, where t_i are the times when A sent to

B and s_i the times when B sent to C. An occurrence of the triple $A \rightarrow B \rightarrow C$ is a pair of times (t_i,s_i) such that $(s_i - t_i) \in [\tau_{min}\ \tau_{max}]$. Thus, we would like to find the maximum number of such pairs which satisfy the causality constraint. It turns out that the causality constraint does not affect the size of the maximum matching, however it is an intuitive constraint in our context.

We now define a slightly more general maximum matching problem: for a pair (t_i, s_i) let $f(t_i, s_i)$ denote the score of the pair. Let M be a matching $\{(t_{i_1}, s_{i_1}), (t_{i_2}, s_{i_2}) \ldots (t_{i_k}, s_{i_k})\}$ of size k. We define the score of M to be

$$Score(M) = \sum_{j=1}^{k} f(t_{i_j}, s_{i_j}).$$

The maximum matching problem is to find a matching with a maximum score. The function $f(t, s)$ captures how likely a message from $B \rightarrow C$ at time s was "caused" by a message from $A \rightarrow B$ at time t. In our case we are using a hard threshold function

$$f(t,s) = f(t-s) = \begin{cases} 1 \text{ if } t - s \in [\tau_{min}, \tau_{max}], \\ 0 \text{ otherwise.} \end{cases}$$

The matching problem for sibling triples is identical with the choice

$$f(t,s) = f(t-s) = \begin{cases} 1 \text{ if } t - s \in [-\delta, \delta], \\ 0 \text{ otherwise.} \end{cases}$$

We can generalize to chains of arbitrary length and siblings of arbitrary width as follows. Consider time lists L_1, L_2, ... ,$L_{\ell-1}$ corresponding to the chain $A_1 \rightarrow A_2 \cdots \rightarrow A_\ell$, where L_i contains the sorted times of communications $A_i \rightarrow A_{i+1}$. An occurrence of this chain is now an $\ell-1$ dimensional matching $\{t_1, t_2, \ldots, t_{\ell-1}\}$ satisfying the constraint $(t_{i+1} - t_i) \in [\tau_{min}\ \tau_{max}]\ \forall\ i = 1,\cdots,\ell - 2$.

The sibling of width k breaks down into two cases - ordered siblings which obey constraints similar to the chain constraints, and unordered siblings. Consider the sibling tree $A_0 \rightarrow A_1, A_2, \cdots A_k$ with corresponding time lists L_1, L_2, ... ,L_k, where L_i contains the times of communications $A_0 \rightarrow A_i$. Once again, an occurrence is a matching $\{t_1, t_2, \ldots, t_k\}$. In the ordered case the constraints are $(t_{i+1} - t_i) \in [-\delta\ \delta]$. This represents A_0 sending communications "simultaneously" to its recipients in the order $A_1, \ldots, A_k$. The unordered sibling tree obeys the stricter constraint $(t_i - t_j) \in [-(k-1)\delta, (k-1)\delta]$, $\forall\ i, j$ pairs, $i \neq j$. This stricter constraint represents A_0 sending communications to its recipients "simultaneously" without any particular order.

Both problems can be solved with a greedy algorithm. The detailed algorithms for arbitrary chains and siblings are given in Figure 4(a). Here we sketch the algorithm for triples. Given two time lists L_1=$\{t_1, t_2, \ldots, t_n\}$ and L_2=$\{s_1, s_2, \ldots, s_m\}$ the idea is to find first valid match (t_{i_1}, s_{i_1}), which is the first pair of times that obey the constraint $(s_{i_1} - t_{i_1}) \in [\tau_{min}\ \tau_{max}]$. Then, recursively find the maximum matching on the remaining sub lists $L_1' = \{t_{i_1+1}, \ldots, t_n\}$ and $L_2' = \{s_{i_1+1}, \ldots, s_m\}$.

The case of general chains and ordered sibling trees is similar. The first valid match is defined similarly. Every pair of entries $t_{L_i} \in L_i$ and $t_{L_{i+1}} \in L_{i+1}$ in the maximum matching must obey the constraint $(t_{L_{i+1}} - t_{L_i}) \in [\tau_{min}\ \tau_{max}]$. To find the first valid match, we begin with the match consisting of the first time in all lists. Denote these times $t_{L_1}, t_{L_2}, \ldots, t_{L_\ell}$. If this match is valid (all consecutive pairs satisfy the constraint) then we are done. Otherwise consider the first consecutive pair to violate this constraint. Suppose it is $(t_{L_i}, t_{L_{i+1}})$; so either $(t_{L_{i+1}} - t_{L_i}) > \tau_{max}$ or $(t_{L_{i+1}} - t_{L_i}) < \tau_{min}$. If $(t_{L_{i+1}} - t_{L_i}) > \tau_{max}$ (t_{L_i} is too small), we advance t_{L_i} to the next entry in the time list L_i; otherwise $(t_{L_{i+1}} - t_{L_i}) < \tau_{min}$ ($t_{L_{i+1}}$ is too small) and we advance $t_{L_{i+1}}$ to the next entry in the time list L_{i+1}. This entire process is repeated until a valid first match is found. An efficient implementation of this algorithm is given in Figure 4. The algorithm for unordered siblings follows a similar logic.

In the algorithms below, we initialize $i = 0; j = 1$ (i, j are time list indices), and $P_1, \ldots, P_n = 0$ (P_k is an index within L_k). Let $t_i = L_i[P_i]$ and $t_j = L_j[P_j]$.

```
1:  Algorithm Chain
2:  while P_k ≤ ||L_k|| − 1, ∀k do
3:     if (t_j − t_i) < τ_min then
4:        P_j ← P_j + 1
5:     else if (t_j − t_i) ∈ [τ_min, τ_max] then
6:        if j = n then
7:           (P_1, ..., P_n) is the next match
8:           P_k ← P_k + 1, ∀k; i ← 0; j ← 1
9:        else
10:          i ← j; j ← j + 1
11:    else
12:       P_i ← P_i + 1; j ← i; i ← i − 1
```

(a)

```
1:  Algorithm Sibling
2:  while P_k ≤ ||L_k|| − 1, ∀k do
3:     if (t_j − t_i) < −(k − 1)δ then
4:        P_j ← P_j + 1
5:     else if (t_j − t_i) > (k − 1)δ, ∀i < j then
6:        P_i ← P_i + 1; j ← i + 1
7:     else
8:        if j = n then
9:           (P_1, ..., P_n) is the next match
10:          P_k ← P_k + 1, ∀k; i ← 0; j ← 1
11:       else
12:          j ← j + 1
```

(b)

Fig. 4. (a) maximum matching algorithm for chains and ordered siblings; (b) maximum matching algorithm for unordered siblings

The next theorem gives the correctness of the algorithms.

Theorem 1. *Algorithm-Chain and Algorithm-Sibling return maximum matchings.*

Proof. By induction. Given a set of time lists $L = (L_1, L_2, \ldots, L_n)$ our algorithm produces a matching $M = (m_1, m_2, \ldots, m_k)$, where each matching m_i is a sequence of n times from each of the n time lists $m_i = (t_1^i, t_2^i, \ldots, t_n^i)$. Let $M^* = (m_1^*, m_2^*, \ldots, m_{k^*}^*)$ be a maximum matching of size k^*. We prove that $k = k^*$ by induction on k^*. We will need the next two lemmas which follow by construction in the Algorithms (we postpone the detailed proofs).

Lemma 1. *If there is a valid matching our algorithm will find one.*

Lemma 2. *Algorithm-Chain and Algorithm-Sibling find an earliest valid matching: let the first valid matching found by either algorithm be* $m_1 = (t_1, t_2, \ldots, t_n)$. *Then for any other valid matching* $m' = (s_1, s_2, \ldots, s_n)$ $t_i \leq s_i \ \forall \ i = 1, \cdots, n$.

If $k^* = 0$, then $k = 0$ as well. If $k^* = 1$, then there exists a valid matching and by Lemma 1 our algorithm will find it.

Suppose that for all sets of time lists for which $k^* = M$, the algorithm finds matchings of size k^*. Now consider a set of time lists $L = (L_1, L_2, \ldots, L_n)$ for which an optimal algorithm produces a matching of size $k^* = M+1$ and consider the first matching in this list (remember that by the causality constraint, the matchings can be ordered). Our algorithm constructs the earliest matching and then recursively processes the remaining lists. By Lemma 2, our first matching is not later than optimal's first matching, so the partial lists remaining after our first matching contain the partial lists after optimal's first matching. This means that the optimal matching for our partial lists must be M. By the induction hypothesis our algorithm finds a matching of size M on these partial lists for a total matching of size $M + 1$.

For a given set of time lists $L = (L_1, L_2, \ldots, L_n)$ as input, where each L_i has a respective size d_i, define the total size of the data as $\|D\| = \sum_{i=1}^{n} d_i$.

Theorem 2. *Algorithm-Chain runs in* $O(\|D\|)$ *time.*

Proof. When looking for a matching, we compare a pair of elements from two time lists. For each comparison, we increment at least once in a time list if the comparison failed. After $n-1$ successful comparisons, we increment in every time list by one. Thus there can be at most $O(\|D\|)$ failed comparisons and $O(\|D\|)$ successful comparisons, since the are $\|D\|$ list advances in total.

Theorem 3. *Algorithm-Sibling runs in* $O(n \cdot \|D\|)$ *time.*

Proof. As in Algorithm-Chain a failed comparison leads to at least one increment, but now $\binom{n}{2}$ successful comparisons are needed before incrementing in every time list. Therefore, in the worst case $O(n^2)$ comparisons lead to $O(n)$ list advances. Since there are at most $\|D\|$ list advances, the maximum number of comparisons is $O(n \cdot \|D\|)$

3.2 Finding all Triples

Assume the data are stored in a vector. Each component in the vector corresponds to a sender id and stores a balanced search tree of receiver lists (indexed by a receiver id). And S is the whole set of the distinct senders. The algorithm for finding chain triples considers sender id s and its list of receivers $\{r_1, r_2, \cdots, r_d\}$. Then for each such receiver r_i that is also a sender, let $\{\rho_1, \rho_2, \cdots, \rho_f\}$ be the receivers to which r_i sent messages. All chains begining with s are of the form $s \rightarrow r_i \rightarrow \rho_j$. This way we can more efficiently enumerate the triples (since we

ignore triples which do not occur). For each sender s we count the frequency of each triple $s \to r_i \to \rho_j$. Assuming all time lists of approximately the same size, the total runtime is $O(d_{max}\|D\|)$, where d_{max} is the size of the longest possible list of receivers and $\|D\|$ is the total dataset size.

4 Statistically Significant Triples

In order to determine the minimum frequency κ that makes a triple statistically significant, we build a statistical model that mimics certain features of the data. In particular we model the inter-arrival time distribution and receiver id probability conditioned on sender id. Using this model, we generate synthetic data and find all randomly occurring triples to determine the threshold frequency κ.

4.1 A Model for the Data

We estimate directly from the data the message inter-arrival time distribution $f(\tau)$, the conditional probability distribution $P(r|s)$, and the marginal distribution $P(s)$ using simple histograms (one for $f(\tau)$, S for $P(r|s)$ and S for $P(s)$, i.e. one conditional and marginal distribution histogram for each sender, where S is the number of senders). One may also model additional features (eg. $P(s|r)$), to obtain more accurate models of the data. One should however bear in mind that the more accurate the model, the closer the random data is to the actual data, hence the less useful the statistical analysis will be - it will simply reproduce the data.

Generating a Synthetic Data Set. Suppose one wishes to generate N messages using $f(\tau)$, $P(r|s)$ and $P(s)$. First we generate N inter-arrival times independently, which specifies the times of the communications. We now must assign sender-receiver pairs to each communication. The senders are selected independently from $P(s)$. We then generate each receiver independently, but conditioned on the sender of that communication, according to $P(r|s)$.

4.2 Determining Significance Threshold

To determine the significance threshold κ, we generate M (as large as possible) synthetic data sets and determine the triples together with their frequencies of occurrence in each synthetic data set. The threshold κ may be selected as the average plus two standard deviations, or (more conservatively) as the maximum frequency of occurrence of a triple.

5 Constructing Larger Graphs using Heuristics

Now we discuss a heuristic method for building larger communication structures, using only statistically significant triples. We will start by introducing the notion of an overlap factor. We will then discuss how the overlap factor is used to build a larger communication graph by finding clusters, and construct the larger communication structures from these clusters.

5.1 Overlap between Triples

Given two statistically significant triples (A, B, C) and (D, E, F) of chain or sibling type, let their maximum matchings occur respectively at the times $M_1 = \{(t_1, s_1), \ldots, (t_k, s_k)\}$ and $M_2 = \{(t'_1, s'_1), \ldots, (t'_p, s'_p)\}$.

We define an overlap weighting function $W(M_1, M_2)$ to capture the degree of coincidence between the matchings M_1 and M_2. The simplest such overlap weighting function is the extent to which the two time intervals of communication overlap. Specifically, $W(M_1, M_2)$ is the percentage overlap between the two intervals $[t_1, s_k]$ and $[t'_1, s'_p]$, if they overlap, and otherwise zero:

$$W(M_1, M_2) = \max \left\{ \frac{\min(s_k, s'_p) - \max(t_1, t'_1)}{\max(s_k, s'_p) - \min(t_1, t'_1)}, 0 \right\}$$

If the *overlap factor* is large, this suggests that both triples are part of the same hidden group. One can define more sophisticated overlap factors to take into account intermittent communication but for our present purpose, the simplest version will suffice.

5.2 The Weighted Overlap Graph and Clustering

We construct a weighted graph by taking all significant triples to be the vertices in the graph. Let M_i be the maximum matching corresponding to vertex (triple) v_i. Then, we define the weight of the edge e_{ij} to be $\omega(e_{ij}) = W(M_i, M_j)$. Thus, we have an undirected complete graph (some weights may be 0). By thresholding the weights, one could obtain a sparse graph. Dense subgraphs in this graph correspond to triples that were all active at about the same time, and are a candidate hidden group. Thus, we want to cluster the graph into dense possibly overlapping subgraphs. Given the triples in a cluster we can build a directed graph which will represent a communication structure within that cluster. The graph built will be constructed to be consistent with all the triples in the cluster, and will usually be a single connected component. If a cluster contains multiple connected components, this may imply the existence of some hidden structure connecting them. Here is an outline of the entire algorithm:

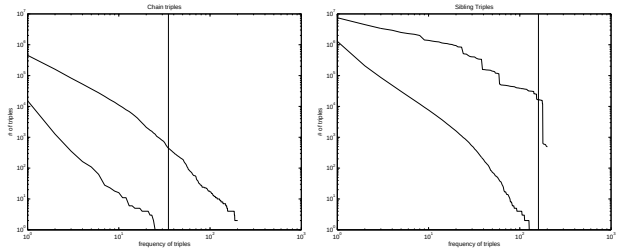
1: Obtain the significant triples.
2: Construct a weighted graph by computing overlap factors between every pair of significant triples.
3: Perform clustering on the weighted graph.
4: Use each cluster to determine a candidate hidden group structure.

The analysis of each step of this algorithm has already been presented except for the clustering. For the clustering, since we allow overlapping clusters, we use the algorithms presented in [5], [6].

6 Experimental Results

6.1 Finding Triples in Enron Data

For our experiments we considered the Enron email corpus. We took τ_{min} to be 1 hour and τ_{max} to be 1 day. Figure 5 compares the number of triples occurring in the data to the number that occur randomly in the synthetically generated data using the model derived from the Enron data. As can be observed, the number of triples in the data by far exceeds the random triples. After some frequency threshold, no random triples of higher frequency appear - i.e., all the triples appearing in the data at this frequency are significant. We used $M = 1000$ data sets to determine the random triple curve in Figure 5. For chains the significance



(a) (b)

Fig. 5. Abundance of triples occurring as a function of frequency of occurrence. (a) chain triples; (b) sibling triples

threshold frequency was $\kappa_{chain} = 35$ and for siblings it was $\kappa_{sibling} = 160$. We used a sliding window of one year to obtain evolving hidden groups. On each window we obtained the significant chains (frequency $> \kappa_{chain}$) and significant siblings (frequency $> \kappa_{sibling}$) and the clusters in the corresponding weighted overlap graph. We use the clusters to build the communication structures and show the evolution of one of the hidden groups in Figure 6.

7 Conclusions

In this paper we give algorithms to find significant chain and sibling triples from streaming communication data. Using a heuristic to build from triples, we find hidden groups of larger sizes. Using a moving window we can track the evolution of the organizational structure as well as hidden group membership.

Our algorithms do not use communication content and do not differentiate between the natures of the hidden groups discovered, i.e. some of the hidden groups may be expected and some may not. Further work, perhaps using human analysts may be needed to identify the trully suspicious groups. This is the

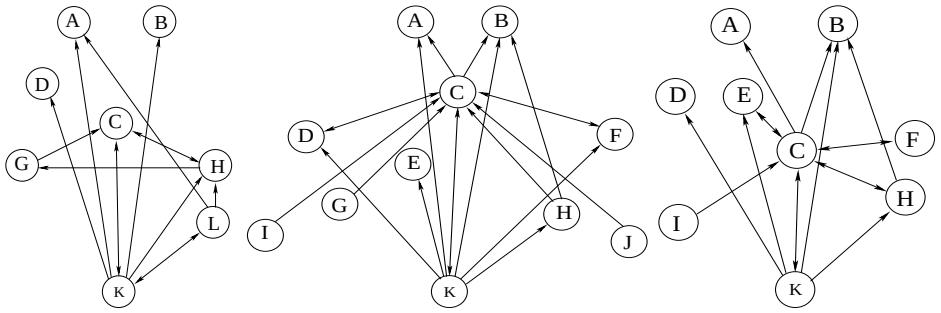


Fig. 6. Evolution of part of the Enron organizational structure from 2000 - 2002 (names have been replaced by letters).

content of future work. Our current algorithms narrow down the set of possible hidden groups that need to be analysed further.

Future work includes exact efficient algorithms to count the frequency of general trees and algorithms to efficiently enumerate all statistically significant general trees of a specified size. In addition, other scoring functions for the matching and overlap weighting functions for the clustering may yield interesting results.

References

1. Monge, P., Contractor, N.: Theories of Communication Networks. Oxford University Press (2002)
2. Magdon-Ismail, M., Goldberg, M., Wallace, W., Siebecker, D.: Locating hidden groups in communication networks using Hidden Markov Models. In: International Conference on Intelligence and Security Informatics (ISI 2003), Tucson, AZ (2003)
3. Baumes, J., Goldberg, M., Magdon-Ismail, M., Wallace, W.: Discovering hidden groups in communication networks. Intelligence and Security Informatics (ISI) (2004) 378–389
4. Baumes, J., Goldberg, M., Magdon-Ismail, M., Wallace, W.: On hidden groups in communication networks. Technical report, TR 05-15, Computer Science Department, Rensselaer Polytechnic Institute (2005)
5. Baumes, J., Goldberg, M., Krishnamoorthy, M., Magdon-Ismail, M., Preston, N.: Finding comminities by clustering a graph into overlapping subgraphs. Proceedings of IADIS Applied Computing (2005) 97–104
6. Baumes, J., Goldberg, M., Magdon-Ismail, M.: Efficient identification of overlapping communities. Intelligence and Security Informatics (ISI) (2005) 27–36