Discovering Hidden Groups in Communication Networks[1]

Jeff BAUMES [a], Mark K. GOLDBERG [a,2], Malik MAGDON-ISMAIL [a], William A. WALLACE [b]

[a] *CS Department, Rensselaer Polytechnic Institute, Troy, NY 12180.*
[b] *DSES Department, Rensselaer Polytechnic Institute, Troy, NY 12180.*

Abstract. This chapter presents statistical and algorithmic approaches to discovering groups of actors that hide their communications within the myriad of background communications in a large communication network. Our approach to discovering hidden groups 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. We distinguish two types of hidden groups: *temporal*, which exhibits repeated communication patterns; and *spatial* which exhibits correlations within a snapshot of communications aggregated over some time interval. We present models and algorithms, together with experiments showing the performance of our algorithms on simulated and real data inputs.

Keywords. statistical communication analysis, terrorist networks, graph clustering, temporal correlation

1. Introduction

1.1. Motivation

Modern communication networks (telephone, email, Internet chatroom, 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 of the communications on the set of actors, as a group. A group of actors may communicate in a structured way while not being forthright in exposing its existence and membership. This chapter develops statistical and algorithmic approaches to discovering such hidden groups.

[1]This article is a reproduction of the article "Identification of Hidden Groups in Communications," by J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace, *Handbook in Information Systems*, Volume 2, pp. 209 - 242; 2007, © Elsevier B. V.

[2]Corresponding Author: Rensselaer Polytechnic Institute; 110 8th street, Troy, N.Y., 12180; USA; Email:goldberg@cs.rpi.edu.

Finding hidden groups on the Internet has become especially important since the September 11, 2001 attacks. The tragic event underlines the need for a tool (a software system) which facilitates 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 of the communication network. The *communication graph* of the network is defined by the set of its actors, as the vertices of the graph, and the set of communications, as the graph's edges. Note that the content of the communications is not used in the definition of the graph. 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 data-sets. The research presented in this chapter 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 to discovering hidden groups 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. Thus, we focus on the discovery of such groups of actors whose communications during the observation time period exhibit statistical *correlations*. We will differentiate between *spatial* and *temporal* correlations, which as we shall see, lead to two different notions of hidden groups.

1.2. Temporal Correlation

One possible instance of temporal correlation is an occurrence of a *repeated communication pattern.* Temporal correlation may emerge as a group of actors are planning some future activity. This planning stage may last for a number of time cycles, and, during each of them, the members of the group need to exchange messages related to the future activity. These message exchanges imply that, with high probability, the subgraph of the communication graph formed by the vertices corresponding to the active members of the group is connected. If this *connectivity* property of the subgraph is repeated during a sufficiently long sequence of cycles, longer than is *expected* for a randomly formed subgraph of the same size, then one can discovers this *higher-than-average* temporal correlation, and hence identify the hidden group.

Thus, in order to detect hidden groups exhibiting temporal correlations, we exploit the non-random nature of their communications as contrasted with the general background communications. We describe efficient algorithms, first appearing in [4,5], which, under certain conditions on the density of the background communications, can efficiently detect such hidden groups. We differentiate between two types of temporally correlated hidden groups: *a trusting, or non-secretive* hidden group, whose members are willing to convey their messages to other hidden group members via actors that are not hidden group members, using these non-hidden group members as "messengers"; and, a *non-trusting, or secretive* hidden group, where all the "sensitive" information that needs to be conveyed among hidden group members uses only other hidden group members as messengers.

Our results reveal those properties of the background network activity and hidden group communication dynamics that make detection of the hidden group easy, as well as those that make it difficult. We find that if the background communications are dense or more structured, then the hidden group is harder to detect. Surprisingly, we also find that

when the hidden group is non-trusting (secretive), it is easier to detect than if it is trusting (non-secretive). Thus, a hidden group which tries to prevent the content of its messages from reaching third parties undermines its operations by becoming easier to detect!

1.3. Spatial Correlation

We use spatial correlation to refer to correlations in the communications of a single communication graph, which represents a snapshot of the communications aggregated over some time interval (in contrast to temporal correlation which refers correlation in the communications over multiple communication graphs which represent successive snapshots of the communications). Spatial correlation of messages initiated by a group of actors in a social network can be identified by a *higher-than-average* total communication level *within* this group. This property does not rely on the content of the messages and is adequately described by the communication graph: the *edge density* of the corresponding set of vertices of the graph is higher than that of the average set. To be able to address a wide variety of applications, we consider a general notion of edge density, which compares the intensity of communications between the actors within a particular set and that between the set and the "outside world." The edge density may be defined in numerous ways depending on the desired characteristics of the discovered groups; our algorithms for discovering groups of higher density (potential hidden groups) are generic with respect to the definition of density. Furthermore, we find only groups which are more dense than any group sufficiently close, which reflects the principle of locality in a social network.

For our numerical experiments, we use two main ideas in defining density: one is the proportion of the number of actual communications to the total number of possible communications; and the other is the ratio of the number of communications within the group to the total number of group communications, including messages to individuals outside the group.

In graph-theoretical terminology, the problem we study is *clustering*. An important implication of our approach is that our algorithms construct clusters that may overlap, *i.e.*, some actors may be assigned to more than one group. While there is much literature in the area of graph clustering, up until very recent work it has mainly focused on a specific sub-case of the problem: graph-partitioning. As opposed to partitioning algorithms, which decompose the network into *disjoint* groups of actors, general clustering allows groups to extend to their natural boundaries by allowing overlap. We discuss prior work in the area of *partitioning*, and present three general *clustering* heuristics, originally described in [2,3]. We refer to these procedures by the names Iterative Scan (IS), Rank Removal (RaRe) and Link Aggregate (LA). We present experimental data that illustrate the efficiency, flexibility, and accuracy of these heuristics.

Searching for both spatial and temporal correlation may be combined to produce a more effective algorithm for the identification of hidden groups. The temporal algorithms may indicate that a large group of individuals are involved in planning some activity. The spatially-correlated algorithm may then be used to cluster this large group into overlapping subgroups, which would correspond to smaller working groups within the larger group.

We present results from the testing of our spatial-hidden group algorithms on a number of real-world graphs, such as newsgroups and email. We analyze the quality of the

groups produced by the clustering algorithms. We also test the algorithms on random graph models in order to determine trends in both runtime and accuracy. One of the interesting experimental discoveries that different implementations of the Iterative Scan algorithm are optimized for different domains of application based on the sparseness (density) of the communication network.

2. Discovering Temporal Correlation

2.1. Literature Review

Identifying temporally correlated hidden groups was initiated in [30] using Hidden Markov models. Here, our underlying methodology is based upon the theory of random graphs [7,22]. We also incorporate some of the prevailing social science theories, such as homophily [32], by incorporating group structure into our model. A more comprehensive model of societal evolution can be found in [20,37]. Other simulation work in the field of computational analysis of social and organizational systems [9,10,36] primarily deals with dynamic models for social network infrastructure, rather than the dynamics of the actual communication behavior, which is the focus of this chapter.

One of the first works analyzing hidden groups is found in [14]. Here, the author studies a number of secret societies, such as a resistance that was formed among prisoners at Auschwitz during the Second World War. The focus, as it is in this paper, was on the structure of such societies, and not on the content of communications. An understanding of a hidden network comes through determining its general pattern and not the details of its specific ties.

The September 11, 2001 terrorist plot spurred much research in the area of discovering hidden groups. Specifically, the research was aimed at understanding the terrorist cells that organized the hijacking. Work has been done to recreate the structure of that network and to analyze it to provide insights on general properties that terrorist groups have. Analyzing their communication structure provides evidence that Mohammed Atta was central to the planning, but that a large percent of the individuals would have needed to be removed in order to render the network inoperable [39]. In "Uncloaking Terrorist Networks" [29], Krebs uses social network measures such as betweenness to identify which individuals were most central to the planning and coordination of the attacks. Krebs has also observed that the network that planned September 11 attempted to hide by making their communications sparse. While these articles provide interesting information on the history of a hidden group, our research uses properties of hidden groups to discover their structure before a planned attack can occur.

There is also work being done in analyzing the theory of networked groups, and how technology is enabling them to become more flexible and challenging to deal with. The hierarchical structure of terrorist groups in the past is giving way to more effective and less organized network structure [35]. Of course, the first step to understanding decentralized groups is to discover them. What follows are some strategies to solve this problem.

2.2. Methodology

A temporally hidden group is a different kind of group from the normally functioning social groups in the society that engage in "random" communications. We define a *tempo-*

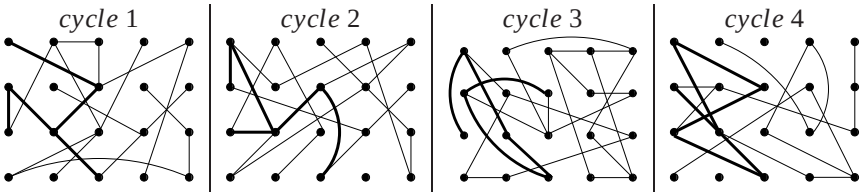


Figure 1. Cyclic representation of the communication.

rally hidden group, or in this section labeled a *hidden group*, as some subset of the actors who are planning or coordinating some activity over time; the hidden group members may also be engaging in other non-planning related communications. The hidden group may be malicious (for example some kind of terrorist group planning a terror attack) or benign (for example a foursome planning their Sunday afternoon golf game). So in this sense, a hidden group is not assumed to be intentionally hiding, but the group activity is initially unknown and masked by the background communications. The hidden group is attempting to coordinate some activity, using the communication network to facilitate the communications between its members. Our task now is to (1) discover specific properties that can be used to find hidden groups; and (2) construct efficient algorithms that utilize those properties. The next steps such as formulation of empirically precise models and further investigation of the properties of hidden groups is beyond the scope of this methodology.

Whether intentional or not, in a normal society, communications will, in general, camouflage the planning related activity of the hidden group. This could occur in any public forum such as a newsgroup or chatrooms, or in private communications such as email messages or phone conversations. However, the planning related activity is exactly the Achilles heel that we will exploit to discover the hidden group: on account of the planning activity, the hidden group members need to stay "connected" with each other during each "communication cycle." To illustrate the general idea, consider the following time evolution of a communication graph for a hypothetical society; here, communications among the hidden group are in bold, and each communication cycle graph represents the communications that took place during an entire time interval. We assume that information must be communicated among *all* hidden group members during one communication cycle (see Figure 1).

Note that the hidden group is connected in each of the communication cycle figures above. We interpret this requirement that the communication subgraph for the hidden group be connected as the requirement that during a single communication cycle, information must have passed (directly or indirectly) from some hidden group member to all the others. If the hidden group subgraph is disconnected, then there is no way that information could have been passed from a member in one of the components to a member in the other, which makes the planning impossible during that cycle. The information need not pass from one hidden group member to every other directly: A message could be passed from A to C via B, where A, B, C are all hidden group members. Strictly speaking, A and C are hidden group members, however B need not be one. We will address this issue more formally in the next section. A hidden group may try to hide its existence by changing its connectivity pattern, or by throwing in "random" communications to non-hidden group members. For example, at some times the hidden group may

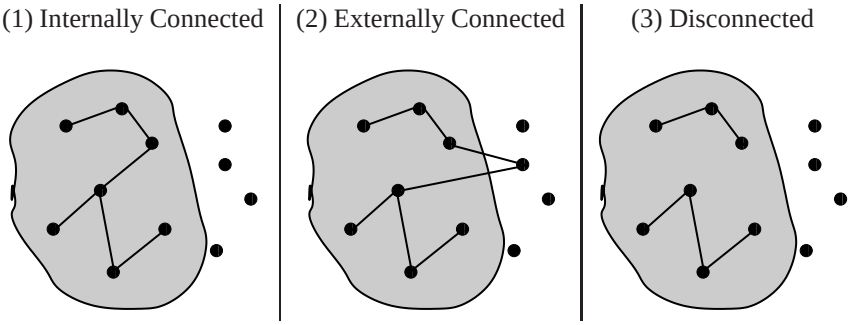


Figure 2. Types of connectivity.

be connected by a tree, and at other times by a cycle. None of these disguises changes the fact that the hidden group is connected, a property we will exploit in our algorithms.

We make the assumption here that the hidden group remains static over the time period when communications are collected. The algorithms described here would still be useful, however, as long as a significant subset of the group remains the same. The algorithms would likely not detect members that joined or left the group, but would discover a "core" group of members.

2.2.1. Trusting vs. Non-Trusting Hidden Groups

Hidden group members may have to pass information to each other indirectly. Suppose that A needs to communicate with B. They may use a number of third parties to do this: $A \rightarrow C_1 \rightarrow \cdots \rightarrow C_k \rightarrow B$. *Trusting* hidden groups are distinguished from *non-trusting* ones by who the third parties C_i may be. In a trusting (or non-secretive) hidden group, the third parties used in a communication may be any actor in the society; thus, the hidden group members (A, B) trust some third-party couriers to deliver a message for them. In doing so, the hidden group is taking the risk that the non-hidden group members C_i have access to the information. For a malicious hidden group, such as a terrorist group, this could be too large a risk, and so we expect that malicious hidden groups will tend to be non-trusting (or secretive). In a non-trusting (secretive) hidden group, *all* the third parties used to deliver a communication *must* themselves be members of the hidden group, *i.e.*, no one else is trusted. The more malicious a hidden group is, the more likely it is to be non-trusting.

Hidden groups that are non-trusting (vs. trusting) need to maintain a higher level of connectivity. We define three notions of connectivity as illustrated by the shaded groups in Figure 2. A group is *internally connected* if a message may be passed between any two group members without the use of outside third parties. In the terminology of Graph Theory, this means that the subgraph induced by the group is connected. A group is *externally connected* if a message may be passed between any two group members, perhaps with the use of outside third parties. In Graph Theory terminology, this means that the group is a subset of a connected set of vertices in the communication graph. For example, in Figure 2 (2) above, a message from A to B would have to use the outside third party C. A group is *disconnected* if it is not externally connected. The following observations are the basis for our algorithms for detecting hidden groups.

(i) Trusting hidden groups are **externally connected** *in every communication cycle.*

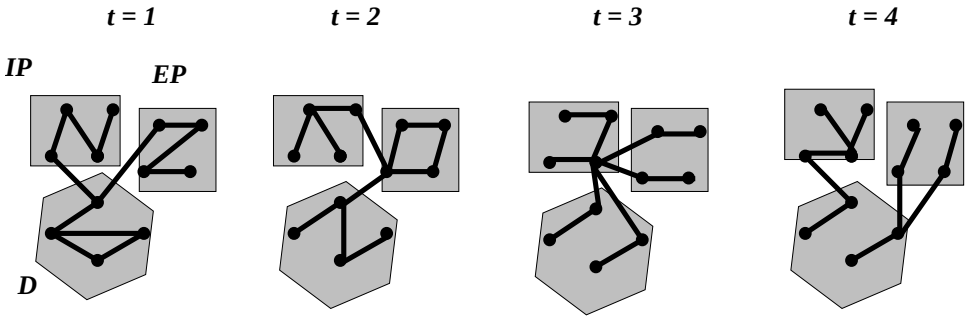


Figure 3. Internally persistent, externally persistent and non-persistent groups. The communication graph during 4 communication cycles is shown. Three groups are highlighted, IP, EP, D. One can easily verify that IP is internally persistent during these 4 communication cycles, and so is a candidate non-trusting hidden group. EP is only internally persistent only for time periods 1 and 2. If we only observed data during this time period, then EP would also be a candidate non-trusting hidden group. However, EP is only externally persistent for all the communication cycles, and hence can only be a candidate for a trusting hidden group. D becomes disconnected during communication cycle 4, and hence is not a candidate hidden group.

(ii) Non-trusting hidden groups are **internally connected** *in every communication cycle.*

We can now state the idea behind our algorithm for detecting a hidden group: a group of actors is *persistent* over communication cycles $1, \ldots, T$ if it is connected in each of the communication graphs corresponding to each cycle. The two variations of the connectivity notion, internal or external, depend on whether we are looking for a non-trusting or trusting hidden group. Our algorithm is intended to discover potential hidden groups by detecting groups that are persistent over a long time period. An example is illustrated in Figure 3.

A hidden group can be hidden from view if, by chance, there are many other persistent subgroups in the society. In fact, it is likely that there will be many persistent subgroups in the society *during any given short time period*. However, these groups will be short-lived on account of the randomness of the society communication graph. Thus we expect our algorithm performance to improve as the observation period increases.

2.2.2. Detecting The Hidden Group

Our ability to detect the hidden group hinges on two things. First, we need an efficient algorithm for identifying maximally persistent components over a time period Π. Second, we need to ensure, with high probability, that over this time period there are no persistent components that arise, by chance, due to the background societal communications. We will construct algorithms to efficiently identify maximal components that are persistent over a time period Π. Given a model for the random background communications, we can determine (through simulation) how long a time period a group of a particular size must be persistent in order to ensure that, with high probability, this persistent component did not arise by chance, due to background communications.

(a) Externally persistent components

1: Ext_Persistent($\{G_t\}_{t=1}^T$, V)
2: //**Input:** Graphs $\{G_t = (E_t, V)\}_{t=1}^T$.
3: //**Output:** A partition $\mathcal{P} = \{V_j\}$ of V.
4: Use DFS to get the connected components $\mathcal{C}_t$ of every G_t;
5: Set $\mathcal{P}_1 = \mathcal{C}_1$ and $\mathcal{P}_t = \{\}$ for $t > 1$;
6: **for** $t = 2$ to T **do**
7: **for** Every set $A \in \mathcal{P}_{t-1}$ **do**
8: Obtain a partition P' of A by intersecting A with every set in $\mathcal{C}_t$;
9: Place P' into $\mathcal{P}_t$;
10: **end for**
11: **end for**
12: **return** $\mathcal{P}_T$;

(b) Internally persistent components

1: Int_Persistent($\{G_t\}_{t=1}^T$, V)
2: //**Input:** Graphs $\{G_t = (E_t, V)\}_{t=1}^T$.
3: //**Output:** A partition $\mathcal{P} = \{V_j\}$ of V.
4: $\{V_i\}_{i=1}^K$ = Ext_Persistent($\{G_t\}_{t=1}^T$, V)
5: **if** $K = 1$, **then**
6: $\mathcal{P} = \{V_1\}$;
7: **else**
8: $\mathcal{P} = \cup_{k=1}^K$ Int_Persistent($\{G_t(U_k)\}_{t=1}^T$, V_k);
9: **end if**
10: **return** $\mathcal{P}$;

Figure 4. Algorithms for detecting persistent components.

2.3. Algorithms

Select Δ to be the smallest time-interval during which it is expected that information is passed among *all* group members. Index the communication cycles (which are consecutive time periods of duration Δ) by $t = 1, 2, \ldots, T$. Thus, the duration over which data is collected is $\Pi = \Delta \cdot T$. The communication data is represented by a series of communication graphs, G_t for $t = 1, 2, \ldots, T$. The vertex set for each communication graph is the set V of all actors.

The input to the algorithm is the collection of communication graphs $\{G_t\}$ with a common set of actors V. The algorithm splits V into persistent components, i.e., components that are connected in every G_t. The notion of connected could be either external or internal, and so we develop two algorithms, Ext_Persistent and Int_Persistent.

Each algorithm develops the partition in an iterative way. If we have only one communication graph G_1, then both the externally and internally persistent components are simply the connected components of G_1. Suppose now that we have one more graph, G_2. The key observation is that two vertices, i, j are in the same external component if and only if they are connected in both G_1 and G_2, i.e., they are in the same component in both G_1 and G_2. Thus, the externally persistent components for the pair G_1, G_2 are exactly the intersections of the connected components in G_1 with those in G_2. This argument clearly generalizes to more than two graphs, and relies on the fundamental property that any subset of an externally connected set is also externally connected. Unfortunately, the same property does not hold for internal connectivity, i.e, a subset of an internally connected set is not guaranteed to be internally connected. However, a minor modification of the externally connected algorithm where one goes back and checks any sets that get decomposed leads to the algorithm for detecting internally persistent components. (Figure 4(b)). The formal details of the algorithms are given in [5].

2.3.1. Analysis

The correctness and computational complexity results of the algorithms given in Figure 4 are stated here. For full detail see [5]. We say that a set A is a *maximal* persistent set (internal or external) if it is persistent, and any other persistent set that contains at least one element of A is a subset of A. Clearly, any two maximal persistent sets must be disjoint, which also follows from the following lemma.

Lemma 1 *If A and B are non-disjoint externally (resp. internally) persistent sets then $A \cup B$ is also externally (resp. internally) persistent.*

Theorem 1 (Correctness of Ext_Persistent**)** *Algorithm* Ext_Persistent *correctly partitions the vertex set V into maximal externally connected components for the input graphs $\{G_t\}_{t=1}^{T}$.*

Let E_t denote the number of edges in G_t, and let E denote the total number of edges in the input, $E = \sum_{t=1}^{T} E_t$. The size of the input is then given by $E + V \cdot T$.

Theorem 2 (Complexity of Ext_Persistent**)** *The computational complexity of Algorithm* Ext_Persistent *is in $O(E + VT)$ (linear in the input size).*

Theorem 3 (Correctness of Int_Persistent**)** *Algorithm* Int_Persistent *correctly partitions the vertex set V into maximal internally connected components.*

Theorem 4 (Complexity of Int_Persistent**)** *The computational complexity of Algorithm* Int_Persistent *is in $O(V \cdot E + V^2 \cdot T)$.*

2.3.2. Statistical Significance of Persistent Components

Let h be the size of the hidden group we wish to detect. Suppose that we find a persistent component of size $\geq h$ over T communication cycles. A natural question is to ask how sure we can be that this is really a hidden group versus a persistent component that happened to arise by chance due to the random background communications.

Let $X(t)$ denote the size of the largest persistent component over the communication cycles $1, \ldots, t$ that arises due to normal societal communications. $X(t)$ is a random variable with some probability distribution, since the communication graph of the society follows a random process. Given a confidence threshold, ϵ, we define the detection time $\tau_\epsilon(h)$ as the time at which, with probability $1-\epsilon$, the largest persistent component arising by chance in the background is smaller than h, *i.e.*,

$$\tau_\epsilon(h) = \min\{t \;\; P X(t) < h \geq 1 - \epsilon\}. \tag{1}$$

Then, if after $\tau_\epsilon(h)$ cycles we observe a persistent component of size $\geq h$, we can claim, with a confidence $1 - \epsilon$, that this did not arise due to the normal functioning of the society, and hence must contain a hidden group. $\tau_\epsilon(h)$ indicates how long we have to wait in order to detect hidden groups of size h. Another useful function is $h_\epsilon(t)$, which is an upper bound for $X(t)$, with high probability $(1 - \epsilon)$, *i.e.*,

$$h_\epsilon(t) = \min\{h \;\; P X(t) < h \geq 1 - \epsilon\}. \tag{2}$$

If, after a given time t, we observe a persistent component with size $\geq h_\epsilon(t)$, then with confidence at least $1-\epsilon$, we can claim it to contain a hidden group. $h_\epsilon(t)$ indicates what sizes hidden group we can detect with only t cycles of observation. The previous approaches to detecting a hidden group assume that we know h or fix a time t at which to make a determination. By slightly modifying the definition of $h_\epsilon(t)$, we can get an even stronger hypothesis test for a hidden group. For any fixed $\delta > 0$, define

$$H_\epsilon(t) = \min\{h \quad PX(t) < h \geq 1 - \tfrac{\delta}{t^{1+\delta}}\epsilon\}. \tag{3}$$

Then one can show that if $X(t) \geq H_\epsilon(t)$ at any time, we have a hidden group with confidence $1-\epsilon$.

Note that the computation of $\tau_\epsilon(h)$ and $h_\epsilon(t)$ constitute a pre-processing of the *society's communication* dynamics. This can be done either from a model (such as the random graph models we have described) or from the true, observed communications over some time period. More importantly, this can be done off-line. For a given realization of the society dynamics, let $T(h) = \min\{t \quad X(t) < h\}$. Some useful heuristics that aid in the computation of $\tau_\epsilon(h)$ and $h_\epsilon(t)$ by simulation can be obtained by assuming that $T(h)$ and $X(t)$ are approximately normally distributed, in which case,

Confidence level	$\tau_\epsilon(h)$	$h_\epsilon(t)$
50%	$ET(h)$	$EX(t)$
84.13%	$ET(h) + \sqrt{VarT(h)}$	$EX(t) + \sqrt{VarX(t)}$
97.72%	$ET(h) + 2\sqrt{VarT(h)}$	$EX(t) + 2\sqrt{VarX(t)}$

(4)

2.4. Random Graphs as Communication Models

Social and information communication networks, *e.g.*, the Internet and WWW, are usually modeled by graphs [33] [9] [10] [36], where the actors of the networks (people, IP-addresses, etc.) are represented by the vertices of the graph, and the connections between the actors are represented by the graph edges. Since we have no *a priori* knowledge regarding who communicates with whom, *i.e.*, how the edges are distributed, it is appropriate to model the communications using a random graph. In this paper, we study hidden group detection in the context of two random graph models for the communication network: uniform random graphs and random graphs with embedded groups. In describing these models, we will use standard graph theory terminology [40], and its extension to *hypergraphs* [6]. In a hypergraph, the concept of an edge is generalized to a *hyperedge* which may join more than two vertices. In addition to these two models, there are other models of random networks, such as the small world model and the preferential attachment model [1]. However, in this work we limited our experiments to the following models, which are also illustrated in Figure 5.

Random Model A simple communication model is one where communications happen at random uniformly among all pairs of actors. Such a communication model can be represented by the random graph model developed and extensively studied by Erdős and Rényi, [11,12,13,7]. In this model, the graph is generated by a random process in which an edge between every pair of vertices is generated independently with a given probability p.

The probability space of graphs generated by such a random process is denoted $G(n, p)$, or sometimes are called the Bernoulli Graphs. We will use the $G(n, p)$ notation throughout this paper.

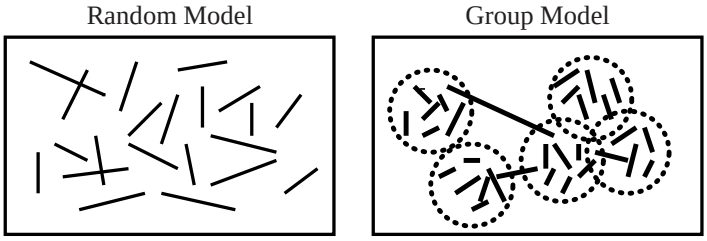


Figure 5. Two random graph models.

Group Model The $G(n, p)$ random graph model may not be a suitable model for large communication networks. Actors tend to communicate more often with certain actors and less frequently with others. In a more realistic model, actors will belong to one or more social groups where communication among group members is more frequent than communication among actors that do not belong to the same group. This leads us to the hypergraph model of the communication network, in which the actors associate themselves into groups. In this paper, we assume that each group is static and contains m actors. While this is a simplification, it serves to illustrate all the essential ideas and results without undue complication. A group of actors is represented by a hyperedge in the graph, and an actor may belong to zero or more hyperedges. The set of all hyperedges represents the structure of the communication network. Since groups tend to be small, it is appropriate to model the communications within a group as a $G(m, p_g)$, where p_g is the probability within the group. We also allow communication between two actors that do not share a group in common; we denote such communications as external. The probability of an external communication is p_e; we further assume that $p_e \ll p_g$ because intra-group communications are much more likely than extra-group communications.

Connectivity of Random Graphs The key idea of our algorithms is based on the following observation. For any subset of actors in a random model network, it is very unlikely that this subset is connected during a "long" consecutive period of time cycles, while a hidden group must stay connected (for its operations) as long as it functions as a group. Thus, we summarize here some results from random graph theory regarding how the connectivity of a $G(n, p)$ depends on n and p, [11,12,13,7]. These results are mostly asymptotic in nature (with respect to n), however, we use them as a guide that remains accurate even for moderately sized n.

Given a graph $G = \{V, E\}$, a subset $S \subseteq V$ of the vertices is connected if there exists a path in G between every pair of vertices in S. G can be partitioned into disjoint *connected components* such that every pair of vertices from the same connected component is connected and every pair of vertices in different connected components is not connected. The size of a component is the number of its vertices; the size of the largest connected component is denoted by $L(G)$.

The remarkable discovery by Erdős and Rényi, usually termed *The Double Jump*, deals with the size of the largest component, and essentially states that $L(G)$ goes through two phase transitions as p increases beyond a critical threshold value. All the results hold asymptotically, with high probability, *i.e.*, with probability tending to 1 when $n \to \infty$:

$$
\begin{array}{c|c}
p = \frac{c}{n} & p = \frac{\ln n}{n} + \frac{x}{n},\ x > 0 \\
\hline
L(G(n,p)) = \begin{cases} O(\ln n) & 0 < c < 1 \\ O(n^{2/3}) & c = 1 \\ \beta(c)n & c > 1,\ \beta(c) < 1 \end{cases} & L(G(n,p)) = n \text{ with prob. } e^{-e^{-x}}
\end{array} \tag{5}
$$

Note that when $x \to \infty$, the graph is connected with probability 1. Since our approach is based on the tenet that a hidden group will display a higher level of connectivity than the background communications, we will only be able to detect the hidden group if the background is not maximally connected, *i.e.*, if $L(G) \neq n$. Thus we expect our ability to detect the hidden group to undergo a phase transition exactly when the background connectivity undergoes a phase transition. For $p =$ constant or $p = d \ln n/n$ with $d > 1$, the graph is asymptotically connected which will make it hard to detect the hidden group. However, when $p =$ constant, connectivity is exponentially more probable than when $p = d \ln n/n$, which will have implications on our algorithms.

2.5. Experiments and Results

In these tests, we simulate societies of sizes $n = 1000$ and 2000. The results for both the random background communication model and the group background communication model are presented in parallel. For each model, multiple time series of graphs are generated for communication cycles $t = 1, 2, \ldots, T$, where $T = 200$. Experiments were run on multiple time series (between five and thirty), and averaged in order to obtain more statistically reliable results. In order to estimate $h_\epsilon(t)$, we estimate $EX(t)$ by taking the sample average of the largest persistent component over communication cycles $1, \ldots, t$. Given h, the time at which the plot of $EX(t)$ drops below h indicates the time at which we can identify a hidden group of size $\geq h$.

We first describe the experiments with the random model $G(n, p)$. The presence of persistently connected components depends on the connectivity of the communication graphs over periods $1, 2, \ldots, T$. When the societal communication graph is connected for almost all cycles, we expect the society to generate many large persistent components. By the results of Erdős and Rényi described in Section 2.4, a phase transition from short-lived to long-lived persistent components will occur at $p = 1/n$ and $p = \ln n/n$. Accordingly, we present the results of the simulations with $p = 1/n$, $p = c \ln n/n$ for $c = 0.9, 1.0, 1.1$, and $p = 1/10$ for $n = 1000$. The rate of decrease of $EX(t)$ is shown in Figure 6. For $p = 1/n$, we expect exponential or super-exponential decay in $EX(t)$ (Figure 6, thin dashed line). This is expected because $L(G)$ is at most a fraction of n. An abrupt phase transition occurs at $p = \ln n/n$ (Figure 6 dot-dashed line). At this point the detection time begins to become large. For constant p (where p does not depend on n, in this case $1/10$), the graph is connected with probability tending to 1, and it becomes essentially impossible to detect a hidden group using our approach without any additional information (Figure 6 thick dashed line). This will occur for any choice of a constant as n becomes large. That is, for any constant $p > 0$, there is an integer N such that if $n > N$ then $G(n, p)$ is connected with high probability, tending to 1.

The parameters of the experiments with the group model are similar to that of the $G(n, p)$-model. We pick the group size m to be equal to 20. Each group is selected independently and uniformly from the entire set of actors; the groups may overlap; and each actor may be a member of zero or more groups. If two members are in the same

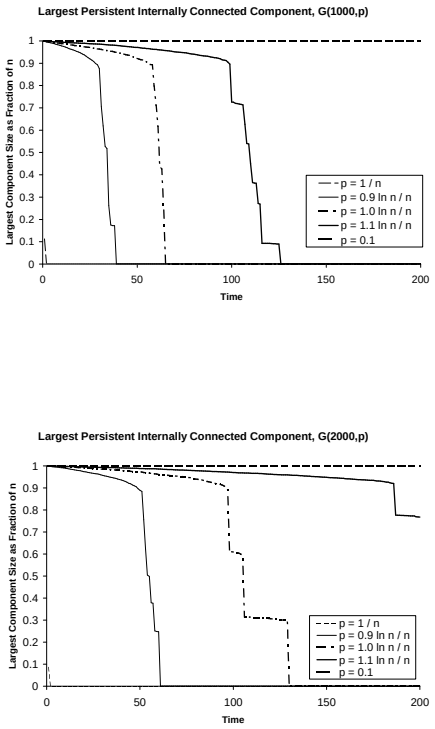
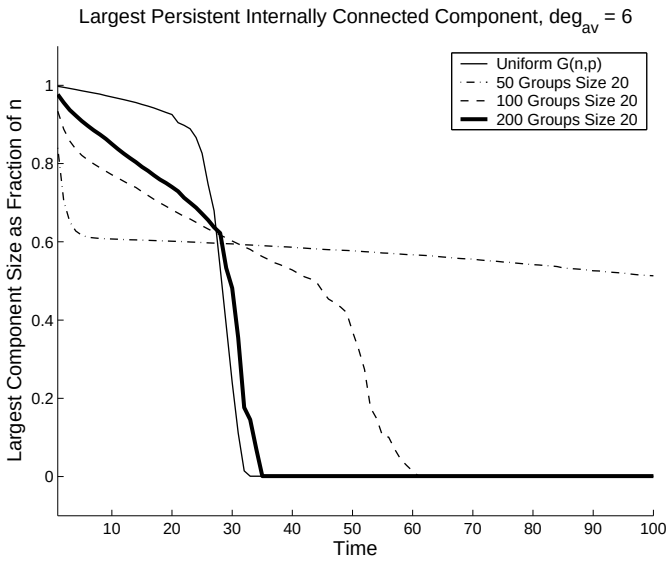


Figure 6. The largest internally persistent component $EX(t)$ for the $G(n, p)$ model with $n = 1000, 2000$. The five lines represent $p = 1/n$, $p = c \ln n/n$ for $c = 0.9, 1.0, 1.1$, and $p = 1/10$. Note the transition at $p = \ln n/n$. This transition becomes more apparent at $n = 2000$. When p is a constant (i.e. does not depend on n; here we used $1/10$), the graph is almost always connected. The results were averaged over a number of runs. The sharp jumps indicate where the largest component quickly jumped from about $0.9n$ to zero in different runs.

group together, the probability that they communicate during a cycle is p_g, otherwise the probability equals p_e. It is intuitive that p_g is significantly bigger than p_e; we picked $p_e = 1/n$, so each actor has about one external communication per time cycle. The values of p_g that we use for the experiments are chosen to achieve a certain average number of communications per actor, thus the effect of a change in the structure of the communication graph may be investigated while keeping the average density of communications constant. The average number of communications per actor (the degree of the actor in the communication graph) is set to six in the experiments. The results do change qualitatively for different choices of average degree. The number of groups g is chosen from $\{50, 100, 200\}$. These cases are compared to the $G(n, p)$ structure with an average of six communications per actor. For the selected values of g, each actor is, on average, in 1, 2 and 4 groups, respectively. When g is 50, an actor is, on average, in approximately one group, and the overlaps of groups are small. However, when g is 200, each actor,



	Group Model			Random Model
g (# of groups)	50	100	200	$G(n, \frac{6}{n})$
$T(1)$	> 100	63	36	32

Figure 7. Times of hidden group discovery for various amounts of group structure; each group is independently generated at random and has 20 actors. In all cases, $n = 1000$, $deg_{av} = 6$, and the group size $m = 20$. Note how, as the number of groups becomes large, the behavior tends toward the $G(n, p)$ case.

on average, is in about 4 groups, so there is a significant amount of overlap between the groups. The goal of our experiments is to see the impact of g on finding hidden groups. Note that as g increases, any given pair of actors tends to belong to at least one group together, so the communication graph tends toward a $G(n, p_g)$ graph.

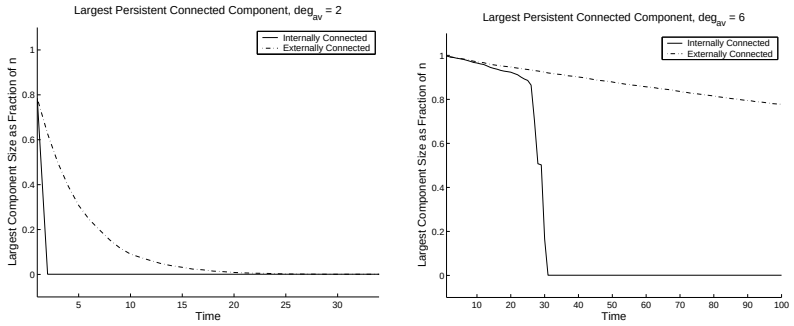
We give a detailed comparison between the society with structure (group model) and the one without (random model) in Figure 7. The table shows $T(1)$, which is the time after which the size of the largest internally persistent component has dropped to 1. This is the time at which any hidden group would be noticed, since the group would persist beyond the time expected in our model.

We have also run similar experiments for detecting trusting groups. The results are shown in Figure 2.5. As the table shows, for the corresponding non-trusting communication model, the trusting group is much harder to detect.

3. Discovering Spatial Correlation

3.1. Literature Review

While an informal definition of the goal of clustering algorithms is straightforward, difficulty arises when formalizing this goal. There are two main approaches to clustering: partitioning and general clustering.



deg_{av}	$T(1)$ for trusting groups	$T(1)$ for non-trusting groups
2	28	2
6	> 100	32

Figure 8. Times of hidden group discovery for non-trusting (internally connected) hidden groups and trusting (externally connected) hidden groups. In all cases the communication graphs are $G(n, p)$ with $n = 1000$.

Partitioning, or hierarchical clustering, is the traditional method of performing clustering. In some circles, clustering and partitioning are synonymous. For example, Kannan, Vempala, and Vetta define clustering as "partitioning into dissimilar groups of similar items" [23]. However, the partitioning approach forces every cluster to be either entirely contained within or entirely disjoint from every other cluster. Partitioning algorithms are useful when the set of objects needs to be broken down into disjoint categories. These categories simplify the network and may be treated as separate entities. Partitioning is used in the fields of VLSI design, finite element methods, and transportation [24].

Many partitioning algorithms attempt to minimize the number of connections between clusters, also called the *cut size* of the partition [27] [26] [21] [24] [25]. The ρ-*separator* metric attempts to balance the sizes of the clusters, while also minimizing the cut size [15]. The *betweenness* metric is used to find a small cut by removing edges that are likely to split the network into components [17,19]. In addition to trying to minimize the cut size, some algorithms attempt to maximize the quality of each cluster. Two metrics used in partitioning which define cluster quality are *expansion* and *conductance* [16,23]. A final metric relates to how well the members of the same cluster are related to the values in eigenvectors of the adjacency matrix of the network [8].

Groups in social networks do not conform to this partitioning approach. For example, in a social network, an individual may belong to numerous groups (*e.g.*, occupational, religious, political, activity). A general clustering algorithm may put the individual into all these clusters, while a partitioning algorithm will only place the individual into one cluster. Classifying an individual as belonging to a single cluster or social group will often miss the full picture of the societal structure. As opposed to partitioning, general clustering allows individuals to belong to many groups at once. General clustering algorithms determine the zero, one or more groups that each actor belongs to, without enforcing a partition structure on the clusters. This complex structure may more directly correspond to real-world clusters. However, permitting overlapping groups is a more complex problem, since each cluster may not be treated as a separate entity. See Figure 9 for a comparison of partitioning and general clustering.

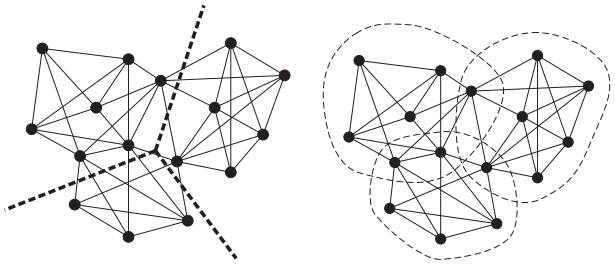


Figure 9. A comparison of a partitioning (a) and a general clustering (b) of the same network.

When clustering a network, there needs to be a definition of what constitutes a "good" cluster. In some sense, members of a cluster need to be "close" to each other, and "far" from the other objects in the network. There are many ways to define the criterion for a valid cluster.

The general clustering problem has been less widely studied than the partitioning problem, however there are some algorithms that exist for discovering a general clustering of a network.

Some algorithms of this type are well suited for web networks [28,18,31]. These algorithms all attempt to find clusters by optimizing a metric referred to as *bicliqueness*. Though often used for partitioning, eigenvector correlation may also be used to discover overlapping clusters [38]. *Local optimality* is a generic technique which can optimize many of the previously mentioned metrics, even metrics originally developed for partitioning. These algorithms have been applied to social networks [2,3]. We present these algorithms in more detail in the following sections.

3.2. Methodology

Let $G = (V, E)$ be a graph whose nodes represent individuals, web pages, etc., and whose edges represent communications, links, etc. The graph may be directed or undirected. We present the definitions for directed graphs, the undirected case being similar. A *graph cluster* C is a set of vertices which can be viewed as a binary vector of length $|V|$ that indicates which nodes are members of C. The set of all graph clusters, $\mathcal{C}$, is the power set of V.

A *weight function*, or *metric* is a function $W \ \mathcal{C} \mapsto \mathbb{R}$ that assigns a weight to a graph cluster. Associated to cluster C, we define three edge sets: $E(C)$, the edges induced by C; $E(C, \overline{C})$, the edges in E *from* C to its complement; $E(\overline{C}, C)$, the edges in E *to* C from its complement. Let $E_{out}(C) = E(C, \overline{C}) + E(\overline{C}, C)$. We define the *internal and external edge intensities*,

$$p_{in}(C) = \frac{E(C)}{|C| \cdot (|C| - 1)}, \qquad p_{ex}(C) = \frac{E_{out}(C)}{2|C| \cdot (n - |C|)} \tag{6}$$

($p_{ex} = 1$ when $|C| = |V|$). We will consider three weight functions: the *internal edge-probability* W_p; the *edge ratio* W_e; and, the *intensity ratio* W_i,

$$W_p(C) = p_{in}(C), \quad W_e(C) = \frac{E(C)}{E(C) + E_{out}(C)}, \quad W_i(C) = \frac{p_{in}(C)}{p_{in}(C) + p_{ex}(C)}. \tag{7}$$

These metrics are measures of how intense the communication within the cluster is, relative to that outside the cluster; they can be efficiently updated locally, i.e. the metric may be updated by knowing only the connectivity of the one node that is added or removed (which improves the efficiency of the algorithms). A *set-difference* function δ is a metric that measures the difference between two clusters C_1, C_2. Two useful set-difference functions are the *Hamming or edit distance* δ_h, and the *percentage non-overlap* δ_p:

$$\delta_h(C_1, C_2) = |(C_1 \cap \overline{C_2}) \cup (\overline{C_1} \cap C_2)|, \qquad \delta_p(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}. \tag{8}$$

The ϵ*-neighborhood* of a cluster $B_\epsilon^\delta(C)$ is the set of clusters that are within ϵ of C with respect to δ, i.e., $B_\epsilon^\delta(C) = \{C' | \delta(C, C') \leq \epsilon\}$. For weight function W, we say that a cluster C^* is ϵ*-locally optimal* if $W(C^*) \geq W(C)$ for all $C \in B_\epsilon^\delta(C^*)$.

We are now ready to formally state our abstraction of the problem of finding overlapping communities in a communication network. The input is a graph G, the communication graph, along with the functions W, δ and ϵ. The output is a set of clusters $\mathcal{O} \subseteq \mathcal{C}$ such that $C \in \mathcal{O}$ *iff* C is ϵ-locally optimal. While our heuristic approaches are easily adapted to different weight and set-difference functions, we will focus on the choices $W = W_e$, $\delta = \delta_h$ and $\epsilon = 1$, referring to the output clusters as locally optimal.

As stated, the problem is NP-hard. In fact, the restriction to $\delta = \delta_h$ and $\epsilon = |V|$ asks to find all the globally optimal clusters according to an arbitrary weight function W, which is well known to be NP-hard. Thus, we present heuristic, efficient (low-order polynomial time) algorithms that output candidate (overlapping) clusters, and then evaluate the quality of the output.

3.3. Algorithms

3.3.1. k-Neighborhood ($k - N$)

$k - N$ is a trivial algorithm that yields overlapping clusters. The clusters are simply the k-neighborhoods of a randomly selected set S of cluster centers. The inputs to this algorithm are k and $|S|$.

*3.3.2. Rank Removal (*RaRe*)*

Algorithm RaRe is based on the assumption that within a communication network, there is a subset of "important" or high-ranking nodes, which do a significant amount of communication. RaRe attempts to identify these nodes and remove them from the graph, in order to disconnect the graph into smaller connected components. The removed node(s) are added to a set R. This process is repeated, until the sizes of the resulting connected components are within a specified range. These connected components can be considered the *core* of each cluster. Next, the vertices in R are considered for addition into one or more of these cores. If a vertex from R is added to more than one cluster, then these clusters now overlap. Note, however, that the cores of each cluster are disjoint, and only communicate with each other through vertices in R.

"Important" or high-ranking nodes are determined by a ranking function ϕ. These are the nodes which are removed at each iteration. We wish to remove nodes that will result in disconnecting the graph as much as possible. One choice is to remove vertices with high degree, corresponding to the choice $\phi_d(v) = deg(v)$. Another approach that we have found to be experimentally better is to rank nodes according to their Page Rank^{TM}, $\phi_p(v)$ [34]. The Page Rank^{TM}of a node is defined implicitly as the solution to the following equation,

$$\phi_p(v) = c \sum_{u,v} \frac{\phi_p(u)}{deg^-(v)} + \frac{(1-c)}{n} \tag{9}$$

Table 1. User specified inputs for Algorithm RaRe.

Input	Description
W	Weight function.
ϕ	Ranking function.
min, max	Minimum and maximum core sizes.
t	Number of high-ranking vertices to remove.

where n is the number of nodes in the graph, $deg^-(v)$ is the out degree of vertex v, and c is a decay factor between 0 and 1. An iterative algorithm to compute $\phi_p(v)$ for all the nodes converges rapidly to the correct value.

Once we have obtained the cores, we must add the vertices in R back into the cores to build up the clusters. Intuitively, a vertex $v \in R$ should be part of any cluster to which it is immediately adjacent, as it would have been part of the core if it were not removed at some step. Also, if we do not take this approach, we run the risk of v not being added to any cluster, which seems counter-intuitive, as v was deemed "important" by the fact that it was at one time added to R. This is therefore the approach which we take. We also add vertices in R to any cluster for which doing so increases the metric W. The algorithm is summarized in Figure 10, and all the user specified inputs are summarized in Table 1.

It is important to note that the initial procedure of removing vertices, though not explicitly attempting to optimize any single metric, does produce somewhat intuitive clusters. The cores that result are mutually disjoint and non-adjacent. Consider a connected component C at iteration i. If C has more vertices than our maximum desired core size max, we remove a set R_i of vertices, where $|R_i| = t$. If the removal of R_i results in disconnecting C into two or more connected components $C_1, C_2 ... C_k$, we have decreased the diameter of $C_1, C_2 ... C_k$ with respect to C, resulting in more compact connected components. If the removal of R_i does not disconnect the graph, we simply repeat the procedure on the remaining graph until it either becomes disconnected or its size is less than max.

As an added performance boost, the ranks may be computed initially, but not recomputed after each iteration. The idea is that if the set R' is being removed, the rank of a vertex v in G will be close to the rank of v in $G - R'$.

*3.3.3. The Link Aggregate Algorithm (*LA*)*

The IS algorithm performs well at discovering communities given a good initial guess, for example when its initial "guesses" are the outputs of another clustering algorithm such as RaRe as opposed to random edges in the communication network. We discuss a different, efficient initialization algorithm here.

RaRe begins by ranking all nodes according to some criterion, such as Page Rank^{TM}[34]. Highly ranked nodes are then removed in groups until small connected components are formed (called the cluster cores). These cores are then expanded by adding each removed node to any cluster whose density is improved by adding it.

While this approach was successful in discovering clusters, its main disadvantage was its inefficiency. This was due in part to the fact that the ranks and connected components need to be recomputed each time a portion of the nodes are removed. The runtime of RaRe is significantly improved when the ranks are computed only once. For the remainder of this paper, RaRe refers to the Rank Removal algorithm with this improvement, unless otherwise stated.

Since the clusters are to be refined by IS, the seed algorithm needs only to find approximate clusters. The IS algorithm will "clean up" the clusters. With this in mind, the new seed algorithm Link Aggregate LA focuses on efficiency, while still capturing good initial clusters. The pseudocode is given in Figure 10. The nodes are ordered according to some criterion, for example

decreasing Page Rank^{TM}, and then processed sequentially according to this ordering. A node is added to any cluster if adding it improves the cluster density. If the node is not added to any cluster, it creates a new cluster. Note, every node is in at least one cluster. Clusters that are too small to be relevant to the particular application can now be dropped. The runtime may be bounded in terms of the number of output clusters C as follows

Theorem 5 *The runtime of* LA *is* $O(|C||E| + |V|)$.

Proof: Let C_i be the set of clusters just before the ith iteration of the loop. The time it takes for the ith iteration is $O(|C_i|deg(v_i))$, where $deg(v_i)$ is the number of edges adjacent to v_i. Each edge adjacent to v_i must be put into two classes for every cluster in C_i: either the other endpoint of the edge is in the cluster or outside it. With this information, the density of the cluster with v_i added may be computed quickly ($O(1)$) and compared to the current density. If $deg(v_i) = 0$, the iteration takes $O(1)$ time. Therefore the total runtime is asymptotically on the order of

$$\sum_{deg(v_i)>0} |C_i|deg(v_i) + \sum_{deg(v_i)=0} 1 \leq \sum_{i=1}^{|V|} |C_i|deg(v_i) + \sum_{i=1}^{|V|} 1 \tag{10}$$

$$\leq \sum_{i=1}^{|V|} |C|deg(v_i) + |V| = 2|C||E| + |V| = O(|C||E| + |V|). \tag{11}$$

Q.E.D

*3.3.4. Iterative Scan (*IS*)*

Algorithm IS explicitly constructs a clusters that is a local maximum w.r.t. a density metric by starting at a "seed" candidate cluster and updating it by adding or deleting one vertex at a time as long as the metric strictly improves. The algorithm stops when no further improvement can be obtained with a single change. This algorithm is given in pseudo-code format in Figure 10. Different local maxima can be obtained by restarting the algorithm at a different seed, or changing the order in which vertices are examined for cluster updating. The algorithm terminates if the addition to C or deletion from C of a single vertex does not increase the weight. During the course of the algorithm, the cluster C follows some sequence, $C_1, C_2, \ldots$, with the property that $W(C_1) < W(C_2) < \cdots$, where all the inequalities are strict. Since the number of possible clusters is finite, the algorithm must terminate when started on *any* seed, and the cluster output will a locally optimal cluster.

The cluster size may be enforced heuristically by incorporating this criterion into the weight function. This is done by adding a penalty for clusters with size outside the desired range. Such an approach will not impose hard boundaries on the cluster size. If the desired range is C_{min}, C_{max}, then a simple penalty function $Pen(C)$, that linearly penalizes deviations from this range is

$$Pen(C) = \max\left\{0, h_1 \cdot \frac{C_{min} - |C|}{C_{min} - 1}, h_2 \cdot \frac{|C| - C_{max}}{|V| - C_{max}}\right\}, \tag{12}$$

where $C_{min}, C_{max}, h_1, h_2$ are user specified parameters. All the user specified inputs are summarized in Table 2.

We emphasize that algorithm IS can be used to improve any seed cluster to a locally optimal one. Instead of building clusters from random edges as a starting point, we can refine clusters, that are output by some other algorithm – these input clusters might be good "starting points", but they may not be locally optimal. IS then refines them to a set of locally optimal clusters.

The original process for IS consisted of iterating through the entire list of nodes over and over until the cluster density cannot be improved. In order to decrease the runtime of IS, we

Table 2. User specified inputs to Algorithm IS.

Parameter	Description
W	Weight function.
δ	Set-difference function ($\delta = \delta_h$ in our implementation).
ϵ	Size of set neighborhood ($\epsilon = 1$ in our implementation).
max_fail	Number of unsuccessful restarts to satisfy stopping condition.
C_{min}, C_{max}	Desired range for cluster size.
h_1, h_2	Penalty for a cluster of size 1 and $\|V\|$.

Table 3. Algorithm performance on real-world graphs. The first entry in each cell is the average value of W_{ad}. The two entries in parentheses are the average number of clusters found and the average number of nodes per cluster. The fourth entry is the runtime of the algorithm in seconds. The e-mail graph represents e-mails among the RPI community on a single day (16,355 nodes). The web graph is a network representing the domain www.cs.rpi.edu/∼magdon (701 nodes). In the newsgroup graph, edges represent responses to posts on the alt.conspiracy newsgroup (4,526 nodes). The Fortune 500 graph is the network connecting companies to members of their board of directors (4,262 nodes).

Algorithm	E-mail	Web
RaRe $\rightarrow$ IS	1.96 (234,9); 148	6.10 (5,8); 0.14
LA $\rightarrow$ IS^2	2.94 (19,25); 305	5.41 (6,19); 0.24
Algorithm	Newsgroup	Fortune 500
RaRe $\rightarrow$ IS	12.39 (5,33); 213	2.30 (104,23); 4.8
LA $\rightarrow$ IS^2	17.94 (6,40); 28	2.37 (288,27); 4.4

make the following observation. The only nodes capable of increasing the cluster's density are the members of the cluster itself (which could be removed) or members of the cluster's immediate neighborhood, defined by those nodes connected to a node inside the cluster. Thus, rather than visiting each node on every iteration, we may skip over all nodes except for those belonging to one of these two groups. If the neighborhood of a cluster is much smaller than the entire graph, this could significantly improve the runtime of the algorithm.

This algorithm provides both a potential decrease and increase in runtime. A decrease occurs when the cluster and its neighborhood are small compared to the number of nodes in the graph. This is the likely case in a sparse graph. In this case, building the neighborhood set N takes a relatively short time compared to the time savings of skipping nodes outside the neighborhood. An increase in runtime may occur when the cluster neighborhood is large. Here, finding the neighborhood is expensive, plus the time savings could be small since few nodes are absent from N. A large cluster in a dense graph could have this property. In this case, placing all nodes in N is preferable.

Taking into account the density of the graph, we may construct N in either of the two methods described here, in order to maximize efficiency in all cases. If the graph is dense, all nodes are placed in N, but if the graph is sparse, the algorithm computes N as the neighborhood of the cluster.

In the experiments that follow, the behavior of IS for sparse graphs is denoted IS, and the behavior for dense graphs is denoted IS^2.

3.4. Experiments and Results

A series of experiments were run in order to compare both the runtime and performance of the new algorithm with its predecessor. In all cases, a seed algorithm was run to obtain initial clusters, then a refinement algorithm was run to obtain the final clusters. The baseline was the seed algorithm

```
procedure RaRe(G, W)
global R ← ∅;
{H_i} are connected components in G;
for all H_i do
  ClusterComponent(H_i);
end for
Initial clusters {C_i} are cluster cores;
for all v ∈ R do
  for all Clusters C_i do
    Add v to cluster C_i if v is adjacent to C_i or
    W(v ∪ C_i) > W(C_i);
  end for
end for
```

```
procedure ClusterComponent(H)
if |V(H)| > max then
  {v_i} are t highest rank nodes in H;
  R ← R ∪ {v_i}; H ← H \ {v_i};
  {F_i} are connected components in H;
  for all F_i do
    ClusterComponent(F_i);
  end for
else if min ≤ |V(H)| ≤ max then
  mark H as a cluster core;
end if
```

```
procedure LA(G, W)
C ← ∅;
Order the vertices v_1, v_2, ..., v_|V|;
for i = 1 to |V| do
  added ← false;
  for all D_j ∈ C do
    if W(D_j ∪ v_i) > W(D_j) then
      D_j ← D_j ∪ v_i; added ← true;
    end if
  end for
  if added = false then
    C ← C ∪ {{v_i}};
  end if
end for
return C;
```

```
procedure IS(seed,G, W)
C ← seed; w ← W(C);
increased ← true;
while increased do
  if G is dense then
    N ← All nodes adjacent to C;
  else
    N ← All nodes in G;
  end if
  for all v ∈ N do
    if v ∈ C then
      C' ← C \ {v};
    else
      C' ← C ∪ {v};
    end if
    if W(C') > W(C) then
      C ← C';
    end if
  end for
  if W(C) = w then
    increased ← false;
  else
    w ← W(C);
  end if
end while
return C;
```

Figure 10. Algorithms Rank Removal (RaRe), Link Aggregate (LA), and Iterative Scan (IS).

RaRe followed by IS. The proposed improvement consists of the seed algorithm LA followed by IS^2. The algorithms were first run on a series of random graphs with average degrees 5, 10, and 15, where the number of nodes range from 1,000 to 45,000. In this simple model, all pairs of communication are equally likely.

All the algorithms take as input a density metric W, and attempt to optimize that metric. In these experiments, the density metric was chosen as W_{ad}, called the *average degree*, which is

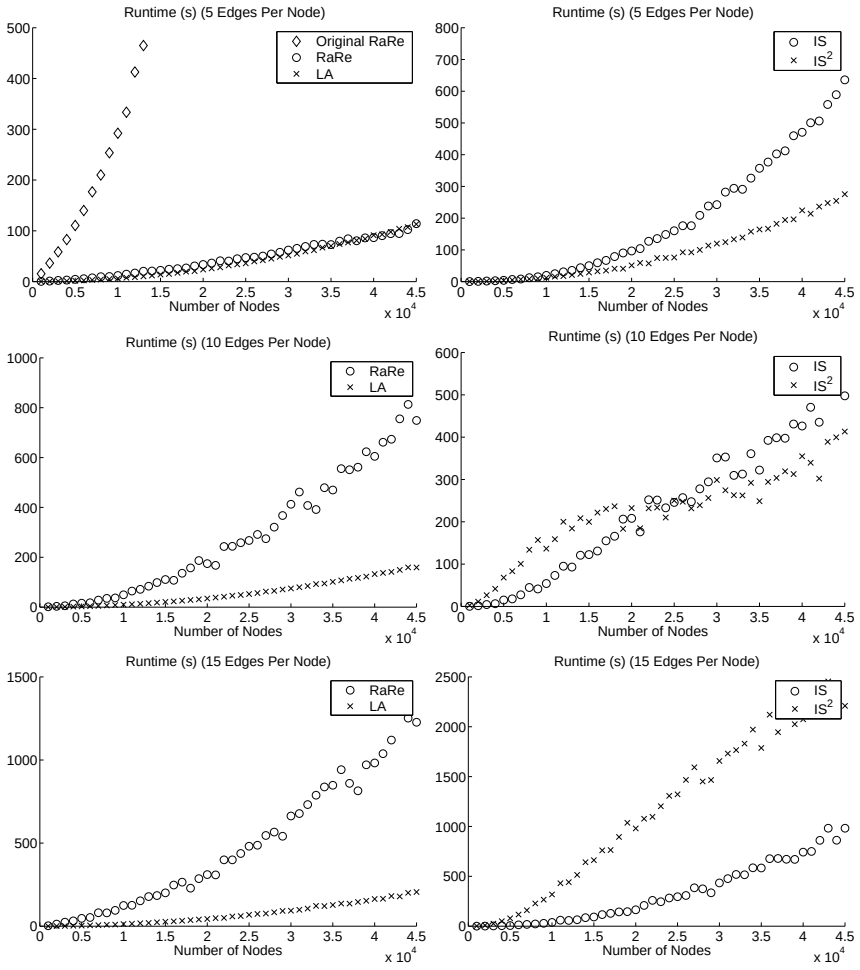


Figure 11. Runtime of the previous algorithm procedures (RaRe and IS) compared to the current procedures (LA and IS^2) with increasing edge density. On the left is a comparison of the initialization procedures RaRe and LA, where LA improves as the edge density increases. On the right is a comparison of the refinement procedures IS and IS^2. As expected, IS^2 results in a decreased runtime for sparse graphs, but its benefits decrease as the number of edges becomes large.

defined for a set of nodes C as

$$W_{ad}(C) = \frac{2|E(C)|}{|C|}, \tag{13}$$

where $E(C)$ is the set of edges with both endpoints in C.

The runtime for the algorithms is presented in Figure 11. The new algorithm remains quadratic, but both the seed algorithm and the refinement algorithm run-times are improved significantly for sparse graphs. In the upper left plot in Figure 11, the original version of RaRe is also plotted, which recalculates the node ranks a number of times, instead of precomputing the ranks a single time. LA is 35 times faster than the original RaRe algorithm and IS^2 is about twice as fast as IS for graphs with five edges per node. The plots on the right demonstrate the tradeoff in

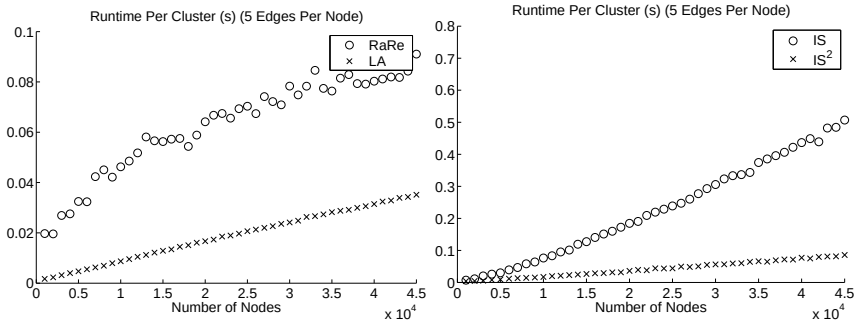


Figure 12. Runtime per cluster of the previous algorithm (RaRe followed by IS) and the current algorithms (LA followed by IS^2). These plots show the algorithms are linear for each cluster found.

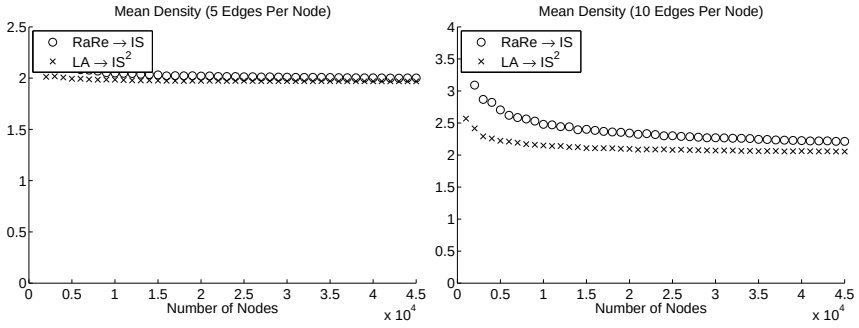


Figure 13. Performance (average density) of the algorithm compared to the previous algorithm.

IS^2 between the time spent computing the cluster neighborhood and the time saved by not needing to examine every node. It appears that the tradeoff is balanced at about 10 edges per node. For graphs that are more dense, the original IS algorithm runs faster, but for less dense graphs, IS^2 is preferable.

Figure 12 shows that the quadratic nature of the algorithm is based on the number of clusters found. When the runtime per cluster found is plotted, the resulting curves are linear.

Runtime is not the only consideration when examining this new algorithm. It is also important that the quality of the clustering is not hindered by these runtime improvements. Figure 13 compares the average density of the clusters found for both the old and improved algorithms. A higher average density indicates a clustering of higher quality. Especially for sparse graphs, the average density is approximately equal in the old and new algorithms, although the older algorithms do show a slightly higher quality in these random graph cases.

Another graph model more relevant to communication networks is the preferential attachment model. This model simulates a network growing in a natural way. Nodes are added one at a time, linking to other nodes in proportion to the degree of the nodes. Therefore, popular nodes get more attention (edges), which is a common phenomenon on the Web and in other real world networks. The resulting graph has many edges concentrated on a few nodes. The algorithms were run on graphs using this model with five links per node, and the number of nodes ranging from 2,000 to 16,000. Figure 14 demonstrates a surprising change in the algorithm RaRe when run on this type of graph. RaRe removes high-ranking nodes, which correspond to the few nodes with very

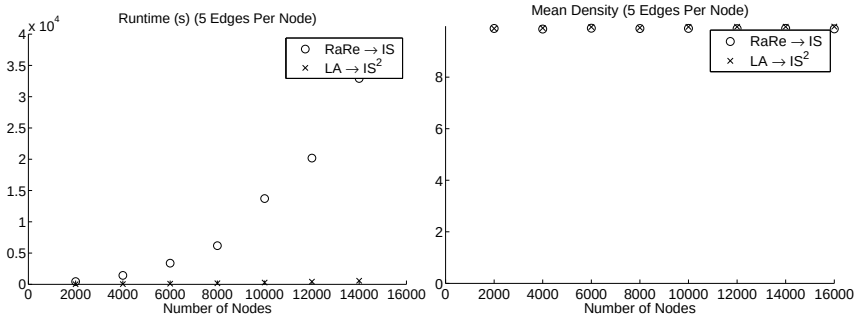


Figure 14. Runtime and performance of the previous algorithm (RaRe followed by IS) and the current algorithm (LA followed by IS^2) for preferential attachment graphs.

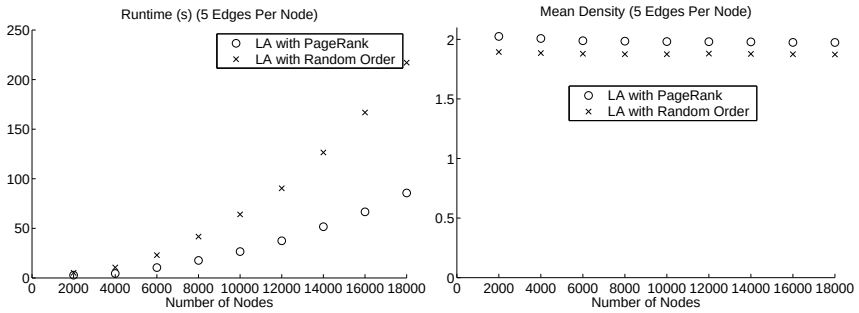


Figure 15. Runtime and performance of LA with two different ordering types.

large degree. When these nodes are added back into clusters, they tend to be adjacent to most all clusters, and it takes a considerable amount of time to iterate through all edges to determine which connect to a given cluster. The algorithm LA, on the other hand, starts by considering high-ranking nodes before many clusters have formed, saving a large amount of time. The plot on the right of Figure 14 shows that the quality of the clusters are not compromised by using the significantly faster new algorithm $LA \rightarrow IS^2$.

Figure 15 confirms that constructing the clusters in order of a ranking such as Page Rank^{TM} yields better results than a random ordering. LA performs better in terms of both runtime and quality. This is a surprising result since the random ordering is obtained much more quickly than the ranking process. However, the first nodes in a random ordering are not likely to be well connected. This will cause many single-node clusters to be formed in the early stages of LA. When high degree nodes are examined, there are many clusters to check whether adding the node will increase the cluster density. This is time consuming. If the nodes are ranked, the high degree nodes will be examined first, when few clusters have been created. These few clusters are likely to attract many nodes without starting a number of new clusters, resulting in the algorithm completing more quickly.

The algorithms were also tested on real-world data. The results are shown in Table 3. For all cases other than the web graph, the new algorithm produced a clustering of higher quality.

4. Conclusion

In this article, we described methods for discovering hidden groups based only on communication data, without the use of communication contents. The algorithms rely on the fact that such groups display correlations in their communication patterns (temporal or spatial). We refer to such groups as hidden because they have not declared themselves as a social entity. Because our algorithms detect hidden groups without analyzing the contents of the messages, they can be viewed as an additional, separate toolkit, different from approaches that are based on interpreting the meaning of the messages. Our algorithms extract structure in the communication network formed by the log of messages; the output groups can further be studied in more detail by an analyst who might take into account the particular form and content of each communication, to get a better overall result. The main advantage is that our algorithms greatly reduce the search space of groups that the analyst will have to look at.

The spatial and temporal correlation algorithms target different types of hidden groups. The temporal hidden group algorithms identify those groups which communicate periodically and are engaged in planning an activity. Our algorithms have been shown to be effective at correctly identifying hidden groups artificially embedded into the background of random communications. Experiments show that as the background communications become more dense, it takes longer to discover the hidden group. A phase transition occurs if the background gets too dense, and the hidden group becomes impossible to discover. However, as the hidden group becomes more structured, the group is easier to detect. In particular, if a hidden group is secretive (non-trusting), and communicates key information only among its members, then the group is actually more readily detectable.

Our approach to the discovery of spatial correlation in communications data is based on the observation that social groups often overlap. This fact rules out the traditional techniques of partitioning and calls for novel procedures for clustering actors into overlapping groups. The families of clustering algorithms described here are able to discover a wide variety of group types based on the clustering metric provided. The algorithms have been shown to be both efficient and accurate at discovering clusters and retaining meaningful overlap between clusters.

Acknowledgments

The research presented here was partially supported by NSF grants 0324947 and 0346341.

References

[1] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74, 2002.

[2] J. Baumes, M. Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston. Finding comminities by clustering a graph into overlapping subgraphs. *Proceedings of IADIS Applied Computing*, pages 97–104, 2005.

[3] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. *Intelligence and Security Informatics (ISI)*, pages 27–36, 2005.

[4] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. Discovering hidden groups in communication networks. *Intelligence and Security Informatics (ISI)*, pages 378–389, 2004.

[5] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. On hidden groups in communication networks. Technical report, TR 05-15, Computer Science Department, Rensselaer Polytechnic Institute, 2005.

[6] C. Berge. *Hypergraphs*. North-Holland, New York, 1978.

[7] Béla Bollobás. *Random Graphs, Second Edition*. Cambridge University Press, new york edition, 2001.

[8] A. Capocci, V. D. P. Servedio, G. Caldarelli, and F. Colaiori. Detecting communities in large networks. *Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 181–188, 2004.

[9] K. Carley and M. Prietula, editors. *Computational Organization Theory*. Lawrence Erlbaum associates, Hillsdale, NJ, 2001.

[10] K. Carley and A. Wallace. Computational organization theory: A new perspective. In S. Gass and C. Harris, editors, *Encyclopedia of Operations Research and Management Science*. Kluwer Academic Publishers, Norwell, MA, 2001.

[11] P. Erdős and A. Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[12] P. Erdős and A. Rényi. On the evolution of random graphs. *Maguar Tud. Acad. Mat. Kutató Int. Kozël*, 5:17–61, 1960.

[13] P. Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Math. Acad. Sci. Hungar.*, 12:261–267, 1961.

[14] Bonnie H. Erickson. Secret societies and social structure. *Social Forces*, 60:188–211, 1981.

[15] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *Siam J. Computing*, 28(6):2187–2214, 1999.

[16] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Clustering methods basen on minimum-cut trees. Technical Report 2002-06, NEC, Princeton, NJ, 2002.

[17] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[18] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[19] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci.*, 99:7821–7826, 2002.

[20] Mark Goldberg, Paul Horn, Malik Magdon-Ismail, Jessie Riposo, David Siebecker, William Wallace, and Bulent Yener. Statistical modeling of social groups on communication networks. In *1st Conf. of the N. Amer. Assoc. for Comp. Social and Organizational Science (NAACSOS)*, PA, June 2003. (electronic proceedings).

[21] Bruce Hendrickson and Robert W. Leland. A multi-level algorithm for partitioning graphs. In *Supercomputing*, 1995.

[22] Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random Graphs*. Series in Discrete Mathematics and Optimization. Wiley, New york, 2000.

[23] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[24] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. 20(1), 1998.

[25] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1), 1998.

[26] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[27] A. Kheyfits. Introduction to clustering algorithms: Hierarchical clustering. *DIMACS Educational Module Series*, 03-1, March 17, 2003.

[28] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: measurements, models, and methods. *Lecture Notes in Computer Science*, 1627, 1999.

[29] Valdis E. Krebs. Uncloaking terrorist networks. *First Monday*, 7 number 4, 2002.

[30] Malik Magdon-Ismail, Mark Goldberg, William Wallace, and David Siebecker. Locating hidden groups in communication networks using Hidden Markov Models. In *Int. Conf. on Intelligence and Security Informatics (ISI)*, pages 126–137, Tucson, AZ, June 2003.

[31] N. Mishra, D. Ron, and R. Swaminathan. Large clusters of web pages. *Workshop on Algorithms and Models for the Web Graph (WAW)*, 2002.

[32] P. Monge and N. Contractor. *Theories of Communication Networks*. Oxford University Press, 2002.

[33] M. E. J. Newman. The structure and function of complex networks. *SIAM Reviews*, 45(2):167–256, June 2003.

[34] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.

[35] David Ronfeldt and John Arquilla. Networks, netwars, and the fight for the future. *First Monday*, 6 number 10, 2001.

[36] Ashish Sanil, David Banks, and Kathleen Carley. Models for evolving fixed node networks: Model fitting and model testing. *Journal oF Mathematical Sociology*, 21(1-2):173–196, 1996.

[37] David Siebecker. A Hidden Markov Model for describing the statistical evolution of social groups over

communication networks. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY 12180, July 2003. Advisor: Malik Magdon-Ismail.
[38] D. B. Skillicorn. Social network analysis via matrix decompositions: al Qaeda.
[39] Thomas A. Stewart. Six degrees of Mohamed Atta. *Business 2.0*, 2 issue 10:63, 2001.
[40] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, U.S.A., 2001.