On Hidden Groups in Communication Networks*

J. Baumes M. Goldberg M. Magdon-Ismail W. Wallace

Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA.
Email: {baumej,goldberg,magdon}@cs.rpi.edu, wallaw@rpi.edu.

Abstract

We describe models and efficient algorithms for detecting groups (communities) functioning in communication networks which use, intentionally or not, the surrounding background communications to disguise their existence. We call such groups *hidden groups*. To detect hidden groups we exploit the non-random nature of their communications as contrasted with the general background communications. We assume that the main goal of the communications of a hidden group in the network is *planning* and *coordination* of their future activity, which by the nature of planning has to involve all, or most members of the group. The implication of this basic assumption is that the communication subgraph induced by the hidden group members is *connected*. We describe efficient algorithms that, under certain conditions, can detect connected subgraphs that remain connected over time.

Our results reveal the 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 differentiate between two types of hidden groups: in a *trusting (or non-secretive)* hidden group, members are willing to communicate messages to other members via non-hidden group members; in a *non-trusting (or secretive)* hidden group, all information that must be passed among hidden group members cannot be passed via non-hidden group members. 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. We also relax our definition of a hidden group to include those groups which persist in any subinterval of communication cycles, and present efficient algorithms and data-types for detecting and querying these groups.

1 Introduction

The tragic events of September 11, 2001 underline the need for a tool which can be used for detecting groups that hide their existence and functionality within a large and complicated communication network, such as the Internet. Public communication forums such

*This research was partially supported by NSF grants 0324947 and 0346341

as Internet newsgroups and chatrooms are an appealing way for members of such groups to communicate because they are easily accessed from almost anywhere in the world, and offer the potential for camouflaging the group's communications among the numerous background communications that are occurring during the same time frame.

In this paper, we view a communication network as a graph. The vertices (nodes) in this graph are the individuals or *actors* of the network, and an edge between two vertices represents a communication between the corresponding actors. This paper will use the terms *vertex* and *actor* interchangeably, within the contexts of graph theory and social networks, respectively. We assume that the communication infrastructure allows for any two actors communicate if they so choose. The communication graph for a society at a particular time step has an edge between two vertices (actors) if they communicated during that time step. The communication graph evolves with time. In general, two actors do not communicate in a deterministic fashion over time. Rather, at "random" times they are communicating, and at other times they are not. Thus, the communication graph that describes the communication dynamics of a network evolves with time according to some stochastic (random) process. The question we ask is:

> *What properties of this randomly evolving communication graph will change if there is a hidden group attempting to camouflage its communications by embedding them into the background communications of the entire communication network?*

Our approach to the problem of reliably detecting hidden group is *non-semantic*. We assume that communication within a hidden group on a public network is usually encrypted in some way, in particular, the communication may not be in English. Furthermore, the semantic information may be either misleading or unavailable or difficult (time-consuming) to process. Thus, we develop a *structural*, rather than semantic way to differentiate between the random evolution of a society's communication graph when it contains a hidden group versus when it does not. The idea behind our technique is based upon the following observation:

> *Normal communications in the network are voluntary and "random," while the communications within a hidden group are focused on achieving certain goals (planning and/or coordination), thus by its nature are likely to be "non-random." Furthermore, a hidden group's communications have to be regular and frequent enough to ensure the succesfull outcome of the planned activity, i.e., a hidden group communicates because it* **has to** *communicate (for planning or coordination purposes).*

Thus, the hidden group communication dynamics will display, out of necessity, certain non-random behavior that differentiates it from normal background communications. Detecting this non-random behavior will help us establish the presence of a hidden group, as well as identify its members.

The property which we use to reveal non-randomness due to the hidden group is based on the connectivity of the communication graph. Specifically, we expect a hidden group to display a connectivity that is more regular than the connectivity of other subsets of the actors. Our analysis and simulations show that, for reasonable communication models of a society, it is possible to efficiently identify the hidden group. This forces a hidden group to face one of two outcomes, both of which are detrimental to its functioning: either continue with its planning or coordination (non-random communication dynamics) and risk being detected, or lower its planning or coordination activity to a level indistinguishable from the random background communication dynamics and risk not achieving its objectives. Our results indicate that there are three major factors that affect our ability to detect a hidden group.

i. *The background communication density in the society.* A higher background communication density, which can be quantified by the average degree of an actor in the communication graph, makes it more difficult to detect hidden groups. More specifically, we observed a phase transition: a hidden group becomes significantly more difficult to detect when the average degree exceeds $\ln n$, where n is the number of actors in the society. Intuitively, a higher background communication density gives more room for the hidden group to hide.

ii. *The presence of dense clusters.* For a given background communication density, we may introduce structure in the communications by forming the actors into groups that communicate together, which results in dense communication clusters in the background communication graph. It is more difficult to detect a hidden group when the society communications are more structured into groups. Intuitively, the hidden group has a certain non-randomness in its structure, and it is easier to differentiate this structure from a random background than from a structured background.

iii. *The type of hidden group.* We differentiate between *trusting (non-secretive)* and *non-trusting (secretive)* groups. Trusting groups allow messages among group members to be delivered by non-group members, whereas non-trusting groups do not. Trusting groups tend to be benign, while non-trusting groups are more likely to be malicious. The surprising result is that it is **easier** to detect non-trusting groups; such groups are undermined by their own paranoia. Intuitively, a trusting group is a less regular structure, where as a non-trusting group is a more compact regular structure, which is easier to differentiate from the random background.

The implications of our results are two-fold. First, we can identify when it is feasible to detect hidden groups. Second, our results allow us to determine how long we must collect communication data to ensure that a hidden group is discovered, under certain assumptions about the rate and nature of the group's communication.

The outline of the remainder of the paper is as follows. In Section 2, we present the work relevant to our field; Section 3 contains the background information on random graph. A formal description of the hidden groups and our algorithms for detecting them are presented in Sections 4 and 5; Section 6 contains the results of our simulations. Possible extensions of our models are discussed in Section 6. The last section contains our conclusions and outlines some further approaches.

2 Related Work

The study of identifying hidden groups was initiated in [14] using Hidden Markov models. Here, our underlying methodology is based upon the theory of random graphs [3, 12]. We also incorporate some of the prevailing social science theories, such as homophily ([15]), by incorporating group structure into our model. A more comprehensive model of societal evolution can be found in [11, 19]. Other simulation work in the field of computational analysis of social and organizational systems [4, 5, 18] primarily deals with dynamic models for social network infrastructure, rather than the dynamics of the actual communication behavior. Our work is novel because we analyze the dynamics of communication intensities in order to detect hidden groups.

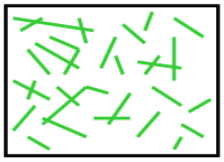
One of the first works analyzing hidden groups is found in [9]. Here, the author studies a number of secret societies, such as a resistance that was formed among prisoners at Auschwitz during the Second World War. The focus, as it is in this paper, was on the structure of such societies, and not on the 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 [20]. In "Uncloaking Terrorist Networks" [13], Krebs uses social network measures such as betweenness that accurately predict 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. This type of group is not amenable to discovery through clustering in the traditional sense, where clusters are defined as dense subsets of the network. 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.

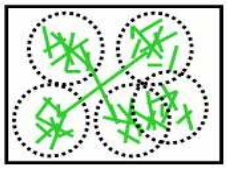
There is also work being done in analyzing the theory of networked groups, and how technology is enabling them to become more flexible and challenging to deal with. The hierarchical structure of terrorist groups in the past is giving way to more effective network structure [17]. Of course, the first step to understanding decentralized groups is to discover them. This work gives some strategies to solve this problem.

3 Random Graphs as Communication Models

Social and information communication networks, such as the Internet and World Wide Web, are usually modeled by graphs (see [16, 4, 5, 18]), 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 (see [21]), and its extension to *hypergraphs* (see [2]). 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.



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, [6, 7, 8, 3]. 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.



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.

3.1 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, [6, 7, 8, 3]. 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 $n \to \infty$:

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

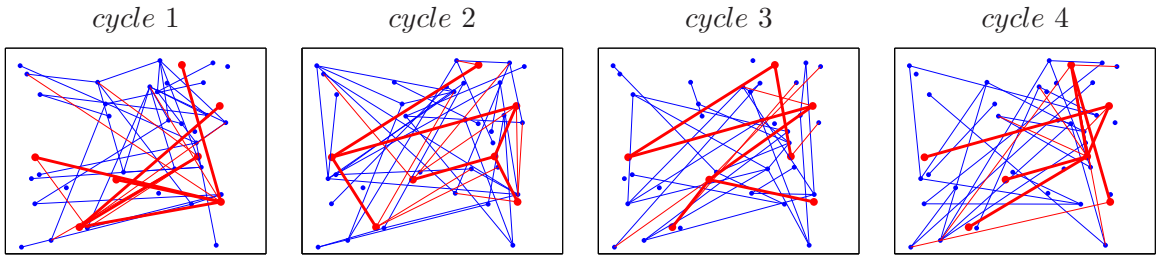
Note that when $x \to \infty$, the graph is connected with probability 1. Since our approach is based on the tenet that a hidden group will display a higher level of connectivity than the background communications, we will only be able to detect the hidden group if the background is not maximally connected, 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.

4 Hidden Groups

A 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 *hidden group* as some subset of the actors who are planning or coordinating some activity; 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, we are liberally using the name "hidden group" in this paper to include all groups involved in planning, though they may not be intentionally attempting to hide their communications. The hidden group is attempting to coordinate some activity, using the communication network to facilitate the communications between its members. Our task here is to (1) to discover specific properties that can be used to find hidden groups; and (2) to 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 paper.

Whether intentional or not, the 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.



Note that the hidden group is connected in each 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 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.

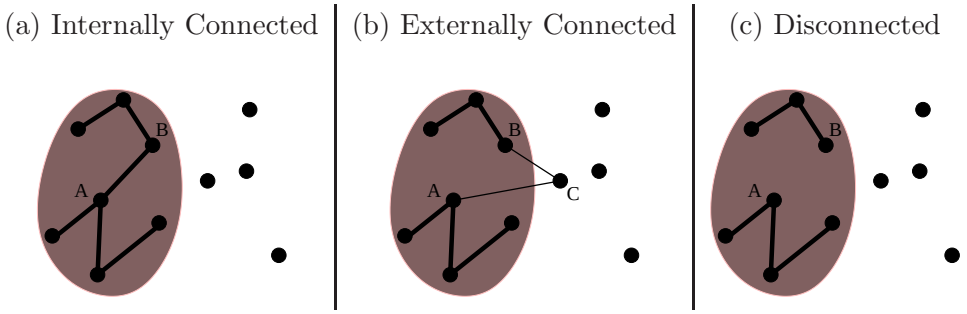
We make the assumption here that the hidden group remains static over the time period when communications are collected. When a hidden is allowed to change, the situation becomes more complex. The algorithms described here would still be useful, however, as long as a significant subset of the group remains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

4.1 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 the following figure.



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 (b) 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.*

(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 enough time period. An example is illustrated in Figure 1.

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 to perform well over a long enough time period.

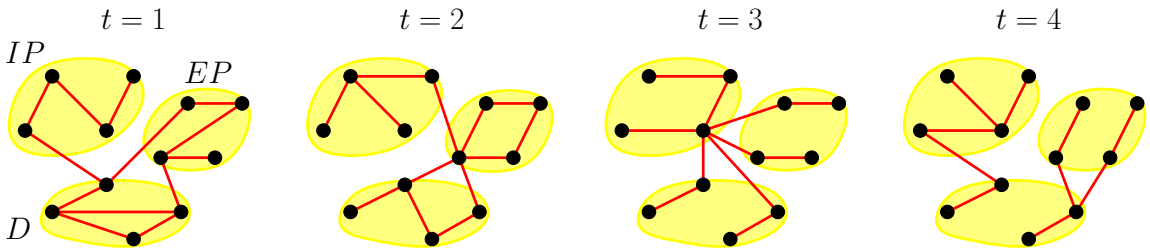


Figure 1: Internally persistent, externally persistent and non-persistent groups. The communication graph during 4 communication cycles is shown. Three groups are highlighted, IP, EP, D. One can easily verify that IP is internally persistent during these 4 communication cycles, and so is a candidate 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.

4.2 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 society 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.

4.2.1 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 output is a partition of 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 the 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 2(b)). We now give the formal details.

Ext_Persistent. Define $\mathcal{C}_t$ as the partitioning of G_t into connected components. Let $\mathcal{P}_t$ denote the partitioning of the vertices into externally persistent components for networks G_1, G_2, ..., G_t, for $t \geq 1$. Clearly $\mathcal{P}_1 = \mathcal{C}_1$. Suppose that $\mathcal{P}_t$ has been constructed. We show how to construct P_{t+1} from $\mathcal{C}_{t+1}$ and $\mathcal{P}_t$ efficiently in time $O(V)$ (step 8 in Algorithm Ext_Persistent in Figure 2 (a)). Suppose that $P_t = \{A_1, A_2, \ldots, A_r\}$ and $\mathcal{C}_{t+1} = \{B_1, B_2, \ldots, B_s\}$. We assume that each vertex x has a pointer to the set $B_j \in \mathcal{C}_{t+1}$, the connected component to which it belongs. We also assume that we have initialized an array of size s containing shadow lists $\{B'_1, B'_2, \ldots, B'_s\}$, all pointing to NULL. Focus on a single element of the current externally persistent partition, $A_i = \{x_1, x_2, \ldots, x_m\}$. Now perform a scan through A_i, and for each member $x \in A_i$, we locate the set B_j to which it belongs and place x in the shadow list B'_j, all this in $O(1)$. Now update $x \in A_i$ to indicate which list it is in. After this scan, A_i has been partitioned, each element of A_i belonging to some B'_j – note that some of the B'_j may be empty. Now scan through A_i one more time, and for each x, if x is in set B'_j, in $O(1)$ time we can access B'_j. If B'_j is empty then we do nothing, otherwise we *empty* the contents of B'_j into another set container, placing it in $\mathcal{P}_{t+1}$. This way, when we process another x that was stored in the same B'_j, the list will already be empty and there will be nothing to do. It is clear that the total work done is $O(|A_i|)$, which includes two scans through A_i, the addition of the x's into the shadow lists, and the emptying of the non-empty shadow lists. Thus, the partitioning of $\mathcal{P}_t$ takes $O(\sum_i |A_i|) = O(V)$ and the initialization of the shadow lists which only occurs once for this entire step is $O(|\mathcal{C}_{t+1}|) = O(V)$. The output is $\mathcal{P}_T$.

(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 2: Algorithms for detecting persistent components.

Int_Persistent. Suppose that $\mathcal{P}_T^{ext}$ is the partition into externally persistent subsets. If $|\mathcal{P}_T^{ext}| = 1$, then this is also a partition into internally persistent subsets. If $|\mathcal{P}_T^{ext}| > 1$, then let U be one of the externally persistent subsets. Let $G_t(U)$ be the subgraph induced by U in G_t. From Lemmas 2, 3 and 4, it follows that any internally persistent subset that overlaps U must be contained in U, and further it must be internally persistent with respect to the sequence of subgraphs $\{G_t(U)\}_{t=1}^T$. These observations lead to the recursive algorithm in Figure 2(b).

4.2.2 Analysis

We now discuss the correctness and computational complexity of the algorithms given in Figure 2. First we will show that Ext_Persistent correctly computes the externally persistent components. 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 $\{G_t\}_{t=1}^T$.*

Proof: We show by induction on t that $\mathcal{P}_t$ satisfies the requirement on $G_1, \ldots, G_t$. Clearly $\mathcal{P}_1$ does so for G_1, which forms the induction basis. For $t > 1$, suppose that $\mathcal{P}_{t-1}$ is a partition into maximal persistent components for $G_1, \ldots, G_{t-1}$. Clearly $\mathcal{P}_t$ is a finer partition of V obtained from $\mathcal{P}_{t-1}$, and so it is also a partition of V. We show that every member of $\mathcal{P}_t$ is externally persistent. Let $A \in \mathcal{P}_t$. Then for some $A' \in \mathcal{P}_{t-1}$ and $B \in \mathcal{C}_t$, $A = A' \cap B$. If $|A| = 1$, then A is persistent. If $|A| > 1$, let i, j be any two vertices in A. Since $i, j \in A'$, they are connected in the graphs $G_1, \ldots, G_{t-1}$. Since $i, j \in B$, they are connected G_t, so i and j are connected in $G_1, \ldots, G_t$, hence A is externally persistent. To show maximality, we show that for any $i \in A$, if some other externally persistent set X contains i, then $X \subseteq A$. Indeed, if $X \not\subseteq A$, then there exists $j \in X$ with $j \notin A$. Since i, j are connected in G_t, $j \in B$. Therefore, $j \notin A'$. Since X is persistent in $G_1, \ldots, G_t$, it is persistent in $G_1, \ldots, G_{t-1}$. But X contains $i \in A'$ (since $i \in A$) and X contains $j \notin A'$, so $X \not\subseteq A'$, which contradicts the maximality of $A' \in \mathcal{P}_{t-1}$. ■

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).*

Proof: Computing $\mathcal{C}_t$, the connected components of G_t, takes time $O(V + E_t)$. As already discussed, to construct $\mathcal{P}_t$ from $\mathcal{P}_{t-1}$ and $\mathcal{C}_t$ takes $O(V)$ time. Therefore, the entire algorithm takes time $O(\sum_t E_t + V) = O(E + VT)$. ■

We now analyze Algorithm Int_Persistent. Let $\mathcal{P}^{ext}$ be the partition into maximal externally persistent components. Clearly any internally persistent partition is also externally persistent, so, by maximality of $\mathcal{P}^{ext}$, every internally persistent partition must be a subset of one of the components in $\mathcal{P}^{ext}$. This remark also applies to a maximal internally persistent component. Let $\mathcal{P}^{int}$ denote the partition into maximal internally persistent components.

Lemma 2 *For every $A^{int} \in \mathcal{P}^{int}$, there exists $A^{ext} \in \mathcal{P}^{ext}$ such that $A^{int} \subseteq A^{ext}$.*

The observation that is the basis for our recursive algorithm Int_Persistent is that if U is an internally persistent subset of the vertices for the input graphs $\{G_t\}_{t=1}^{T}$, then U is internally persistent for the induced subgraphs $\{G_t(U)\}_{t=1}^{T}$. Together with the observation that if $U' \subseteq U$, then $G_t(U')$ is a subgraph of $G_t(U)$, we have

Lemma 3 *Let $A^{int} \in \mathcal{P}^{int}$, $A^{ext} \in \mathcal{P}^{ext}$ with $A^{int} \subseteq A^{ext}$. Then A^{int} is maximally internally persistent for the induced subgraphs $G_t(A^{ext})$.*

Combining Lemma 2 with Lemma 3, we now immediately obtain the following result.

Lemma 4 *Let $A^{int} \in \mathcal{P}^{int}$, $A^{ext} \in \mathcal{P}^{ext}$ with $A^{int} \subseteq A^{ext}$. Let $\mathcal{Q}^{ext}$ be a maximal externally persistent partition of A^{ext} for the induced subgraphs $G_t(A^{ext})$. Then, $A^{int} \subseteq B^{ext}$ for some $B^{ext} \in \mathcal{Q}^{ext}$.*

The direct implementation of Lemma 3 leads to our algorithm Int_Persistent in Figure 2. We need one final result that will imply the correctness of algorithm Int_Persistent.

Lemma 5 *Given a sequence of graphs $\{H_t\}_{t=1}^T$ with common vertex set U, let A^{ext} be a maximal externally persistent subset of U. A^{ext} is also internally persistent if and only if the partition of A^{ext} into maximal externally persistent components with respect to the induced subgraphs $\{H_t(A^{ext})\}_{t=1}^T$ has only one component. Further, in this case, A^{ext} is also a maximal internally persistent subset of U.*

Proof: The maximality follows from the maximality of A^{ext}, since any internally persistent subset has to be a subset of a maximal externally persistent subset (Lemma 2). Certainly if A^{ext} is internally persistent, then the partition of A^{ext} into maximal externally persistent components with respect to the induced subgraphs $\{H_t(A^{ext})\}_{t=1}^T$ will have only one component (Lemma 3). Suppose the partition of A^{ext} into maximal externally persistent components with respect to the induced subgraphs $\{H_t(A^{ext})\}_{t=1}^T$ has only one component. This component must be the entire set A^{ext}, therefore every $\{H_t(A^{ext})\}_{t=1}^T$ is connected. But this is exactly the requirement for A^{ext} to be internally persistent with respect to $\{H_t\}_{t=1}^T$ ■

Theorem 1 together with Lemmas 2, 3, 4 and 5 imply the following theorem.

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

Proof: We trace the algorithm as it outputs a component A of the internally persistent partition. We represent this trace as, $V = A_0 \rightarrow A_1 \rightarrow \cdots \rightarrow A_k = A \rightarrow A$, which indicates that we start with V, take one of its maximal externally persistent components, A_1. We then take a maximal externally persistent component of A_1 with respect to its induced subgraphs, A_2. This process continues till A_k, which is A. Finally we construct the maximal externally persistent partition of A_k with respect to its induced subgraphs, and this partition contains only one component, which we return as A. Note that A is the first (and only) repetition in this sequence of sets. We say that the depth of A is k. By Lemma 5, A is internally persistent. It remains to prove maximality. We use induction on the depth k. For any set of graphs, $\{H_t\}$, all the components output at depth 0 are maximal. This is clear because the only possibility is that the component is the entire vertex set. Suppose that for any set of graphs, $\{H_t\}$, all the components output at depth $\leq k$ are maximal. We now consider output A of depth $k+1$, and suppose that it is not maximal. Then, there is some other internally persistent set A' that contains $i \in A$ and $j \notin A$. Therefore,

$A'' = A \cup A'$ is internally persistent (Lemma 1). Since A'' must be a subset of some maximal externally persistent component (Lemma 2), it must be that $A'' \subseteq A_1$. Therefore, we can restrict our attention to $\{H_t(A_1)\}$, since A'' is internally persistent w.r.t. $\{H_t(A_1)\}$. A will be a depth k internally persistent component output by the algorithm with respect to the graphs $\{H_t(A_1)\}$. However, A will not be maximal (since A' is internally persistent and $A' \not\subseteq A$), which contradicts the induction hypothesis. ■

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

Proof: The maximum depth (defined in the proof of Theorem 3) of a component is $V-1$, since the maximum size of a component drops by at least 1 for every unit depth increase. In the worst case, at every depth, the algorithm runs Ext_Persistent on subsets that total the whole set V. Since the run time of Ext_Persistent is linear, summing over all these subsets gives a total computation at a given depth of $O(E + V \cdot T)$, and the bound on the depth gives a run time of $O(V \cdot E + V^2 \cdot T)$. The formal proof is by induction on V.

Let $\tau(E, V, T)$ be the maximum runtime on an input with these parameters. We claim that $\tau(E, V, T) \leq \alpha(V \cdot E + V^2 \cdot T)$, for some α. If $V = 1$, then the claim is trivial. Suppose that for some $V > 1$, the claim holds for all smaller V, and consider any problem with vertex set size V. If the number of maximal externally persistent components, K, is 1, then the algorithm runs Ext_Persistent once, in time $\leq \beta(E + V \cdot T)$. If $K > 1$, then the algorithm runs Ext_Persistent ($\leq \beta(E+V \cdot T)$); needs to compute all the induced subgraphs, which can be done in linear time all at once for all the externally persistent components ($\leq \gamma(E + V \cdot T)$); and finally runs Ext_Persistent on all the components. Suppose that component k defines a problem of size $E(k), V(k)$. Since $V(k) < V$, we have

$$\begin{aligned} \tau(E, V, T) &\leq \sum_{k=1}^{K} \tau(E(k), V(k), T) + (\beta + \gamma)(E + VT), \\ &\leq \alpha \sum_{k=1}^{K} \big(V(k) \cdot E(k) + V^2(k)T\big) + (\beta + \gamma)(E + VT). \end{aligned}$$

Note that

$$\sum_k V(k) \cdot E(k) = \sum_k (V(k) \cdot (E - (E - E(k))) = V \cdot E - \sum_k V(k) \cdot (E - E(k)).$$

Let F denote the residual edges that are not included in any $E(k)$. Since $K \geq 2$, $E - E(k) \geq E(k+1) + F$ for $k < K$ and $E - E(K) \geq E(1) + F$. Since $V(k) \geq 1$, we have that $\sum_k V(k) \cdot (E - E(k)) \geq \sum_k (E(k) + F) = E + (K-1)F \geq E$. We conclude that

$$\sum_k V(k) \cdot E(k) \leq V \cdot E - E.$$

Because $V(k) \geq 1$, the identity $V^2 = \sum_k V^2(k) + \sum_{i \neq j} V(i) \cdot V(j)$ gives $V^2 \geq \sum_k V^2(k) + V$, hence

$$\sum_k V^2(k) \leq V^2 - V.$$

Putting all this together, we find that

$$\tau(E, V, T) \leq \alpha(V \cdot E + V^2 \cdot T) + (\beta + \gamma - \alpha)(E + V \cdot T).$$

For any $\alpha > (\beta + \gamma)$, the claim follows by induction. ■

4.2.3 Statistical Significance of Persistence 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 high probability governed by ϵ, the largest persistent component arising by chance in the background is smaller than h, *i.e.*,

$$\tau_\epsilon(h) = \min\{t : P[X(t) < h] \geq 1 - \epsilon\}.$$

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, *i.e.*,

$$h_\epsilon(t) = \min\{h : P[X(t) < h] \geq 1 - \epsilon\}.$$

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

$$H_\epsilon(t) = \min\{h : P[X(t) < h] \geq 1 - \tfrac{\delta}{t^{1+\delta}}\epsilon\}.$$

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 $T(h) = \min\{t : X(t) < h\}$. 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%	$E[T(h)]$	$E[X(t)]$
84.13%	$E[T(h)] + \sqrt{Var[T(h)]}$	$E[X(t)] + \sqrt{Var[X(t)]}$
97.72%	$E[T(h)] + 2\sqrt{Var[T(h)]}$	$E[X(t)] + 2\sqrt{Var[X(t)]}$

5 Experiments

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 $E[X(t)]$ 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 $E[X(t)]$ 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 3, 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 $E[X(t)]$ is shown in Figure 3. For $p = 1/n$, we expect exponential or super-exponential decay in $E[X(t)]$ (Figure 3, 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 3 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 3 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

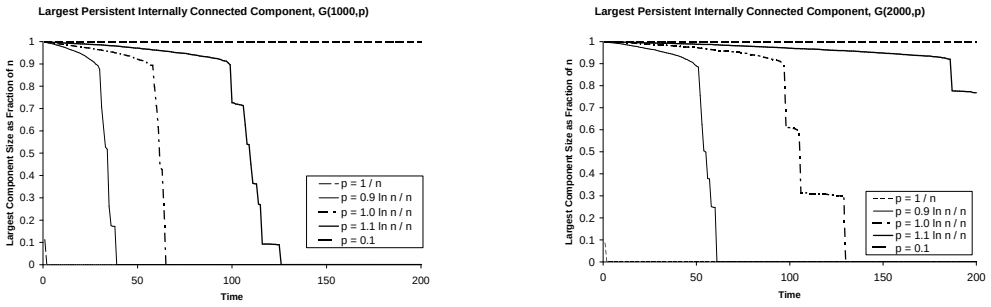
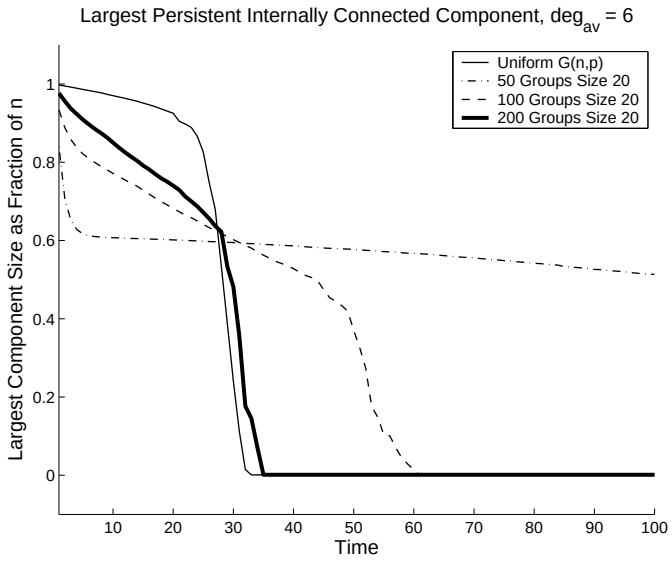


Figure 3: The largest internally persistent component $E[X(t)]$ for the $G(n,p)$ model with $n = 1000, 2000$. The five lines represent $p = 1/n$, $p = c \ln n/n$ for $c = 0.9, 1.0, 1.1$, and $p = 1/10$. Note the transition at $p = \ln n/n$. This transition becomes more apparent at $n = 2000$. When p 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.

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



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

Figure 4: 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.

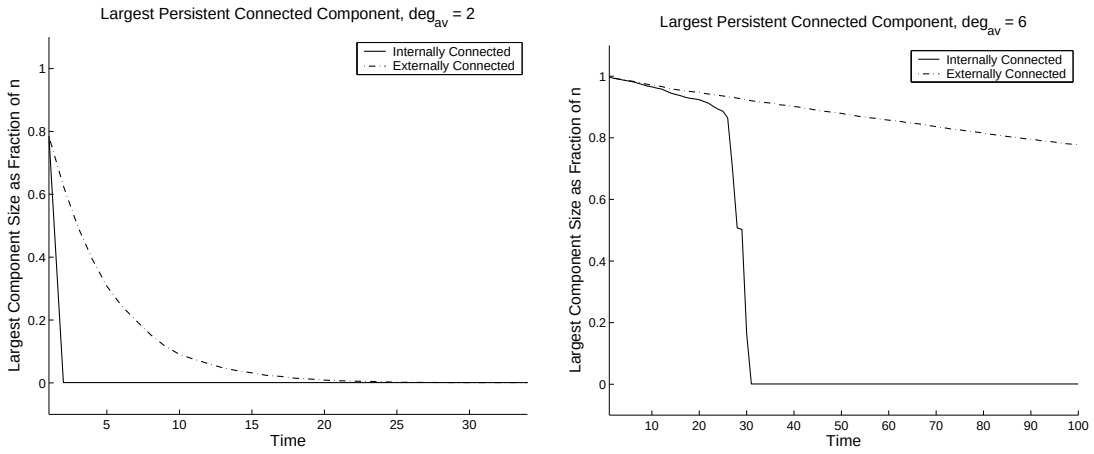
a $G(n, p_g)$ graph.

We give a detailed comparison between the society with structure (group model) and the one without (random model) in Figure 4. 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 5. As the table shows, for the corresponding non-trusting communication model, the trusting group is much harder to detect.

6 Searching for Hidden Groups in Unknown Intervals

Until now we have assumed that the hidden group is persistent throughout all cycles $1, 2, \ldots, T$. Suppose, however, that the hidden group begins its planning at some cycle i and ends planning at another cycle j, where both i and j are unknown. The goal then



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

Figure 5: 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 $n = 1000$.

is to be able to discover hidden groups which plan in *any* subinterval cycles $[i, j]$ with $1 \leq i \leq j \leq T$. Let us denote the partition containing all hidden groups persistent in the interval $[i, j]$ as $\mathcal{P}(i, j)$. We present an algorithm efficient in both time and memory which produces a database of all hidden groups persistent in any interval. We then discuss possible queries on that database to extract useful information.

6.1 Unknown Interval Algorithm

One simple approach to this problem is to call the original hidden group algorithm for each interval $\{G_t\}_{t=i}^{j}$. Since there are $T(T-1)/2$ intervals, this is a time-consuming process. This may be improved by taking into account an observation about the nature of partitions in subintervals. Specifically, if an interval $[i, j]$ is contained within the interval $[k, l]$, then the partition $\mathcal{P}(i, j)$ is a *refinement* of $\mathcal{P}(k, l)$. A partition A is said to be a refinement of partition B if every set in B is either equal to a set in A or is equal to a union of two or more sets in A. Alternatively stated, extending the interval being considered will further break down the hidden groups which are persistent.

The resulting efficient algorithm is based on dynamic programming, where partitions for smaller intervals are used for developing the partitions for larger intervals. The algorithm begins by computing the hidden groups which are persistent for one time cycle, i.e. the connected components of each graph G_i. Now assume that all hidden groups persistent

for k cycles have been computed. Suppose we now wish to determine the hidden groups persistent in the range $[i, i+k]$, which contains $k+1$ cycles. By our observation, we know that the partition $\mathcal{P}(i, i+k)$ is a refinement of both $\mathcal{P}(i, i+k-1)$ and $\mathcal{P}(i+1, i+k)$. Each of these partitions are of length k, hence they have already been computed. So we already know that $\mathcal{P}(i, i+k)$ is equal to or a refinement of the *intersection* of $\mathcal{P}(i, i+k-1)$ and $\mathcal{P}(i+1, i+k)$, where the intersection C of two partitions A and B is the refinement partition of A and B that contains the minimum number of sets. The original hidden group algorithm then possibly refines the sets in this intersection. Since these sets are often much smaller than what would be used by the original algorithm, the procedure will complete with fewer recursive iterations. Additionally, if the same set is found in both $\mathcal{P}(i, i+k-1)$ and $\mathcal{P}(i+1, i+k)$, it must be persistent in the full range $[i, i+k]$, so it may be immediately added to $\mathcal{P}(i, i+k)$.

Unknown_Interval($\{G_t\}_{t=1}^{T}$)
//**Input:** Graphs $\{G_t\}_{t=1}^{T}$.
//**Output:** An array of partitions containing each $\mathcal{P}(i,j)$ for all $1 \leq i \leq j \leq T$.
Use DFS to obtain the connected components $\mathcal{P}(t,t)$ for every G_t.
for $k = 1$ to $T-1$ **do**
  for $i = 1$ to $T-k$ **do**
    $\mathcal{P}(i, i+k) = \emptyset$;
    Compute the intersection $\mathcal{C}$ of $\mathcal{P}(i, i+k-1)$ and $\mathcal{P}(i+1, i+k)$;
    for $C_j \in \mathcal{C}$ **do**
      if $C_j \in \mathcal{P}(i, i+k-1)$ and $C_j \in \mathcal{P}(i+1, i+k)$ **then**
        $\mathcal{P}(i, i+k) = \mathcal{P}(i, i+k) \cup \{C_j\}$;
      else
        $\mathcal{P}(i, i+k) = \mathcal{P}(i, i+k) \cup$ Int_Persistent($\{G_t\}_{t=i}^{i+k}, C_j$);
      end if
    end for
  end for
end for
return $\mathcal{P}$;

6.2 Set Lattice Data Structure

The unknown interval algorithm must store the hidden groups found in each of the $T(T-1)/2$ intervals. If the membership of all hidden groups is stored for all the possible intervals, this requires $\Theta(|V|)$ space for each interval. This is due to the fact that for each interval, the hidden groups constitute a partition of the vertices. The total space required by explicitly storing all group memberships is therefore $\Theta(|V| * T(T-1)/2)$.

The storage required may be reduced by making use of the observation that smaller intervals produce coarser partitions, as was used in developing the unknown interval al-

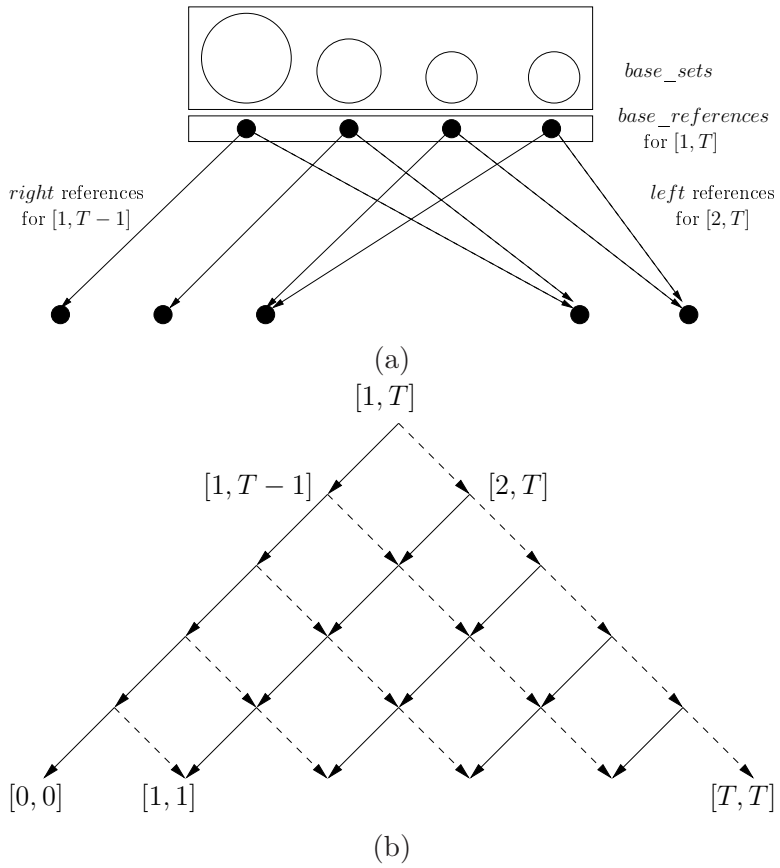


Figure 6: (a) The structure of the base of a set lattice. The most refined partition $\mathcal{P}(1,T)$ is stored in *base_sets*, while subinterval partitions reference the sets of larger intervals. (b) The overall structure of the set lattice data-type. The solid arrows indicate left references while the dotted arrows indicate right references. Each arrow actually represents a group of references from each set in the upper interval to its superset in the lower interval.

gorithm. By taking advantage of this fact, we may construct a more efficient method for storing all partitions in a structure called a *set lattice*. A set lattice only stores the membership of the base partition $\mathcal{P}(1,T)$, or the hidden groups which persist over the entire time range. This base partition is the most refined partition. Each set in the partition for a smaller interval may then be defined by the sets in an extended interval which make up that set. For each set in the partition $\mathcal{P}(i,j)$, we store the partition set to which it belongs in $\mathcal{P}(i,j-1)$ in addition to the set to which it belongs in $\mathcal{P}(i+1,j)$. The reference to the set in $\mathcal{P}(i,j-1)$ is called the *left reference* while the reference to the set in $\mathcal{P}(i+1,j)$ is called the *right reference*. Figure 6 shows how the references connect the sets of neighboring partitions.

In order to build the set lattice structure incrementally, we introduce the Combine proce-

dure. This algorithm takes as input two set lattices, representing the left and right children of the combined lattice, along with the partition of the base lattice. In order to generate a lattice with base interval $[i, j]$, Combine takes as input the lattices based at $[i+1, j]$ and $[i, j-1]$, along with the partition $\mathcal{P}(i, j)$, and generates the proper left and right references from the sets in $\mathcal{P}(i, j)$ to the corresponding supersets in the two subintervals.

Combine$(M, N, \mathcal{P})$
//**Input:** Set lattices M and N, partition $\mathcal{P}$.
//**Output:** Combined set lattice L.
$L.base\_sets = \mathcal{P}$;
for $P_i \in \mathcal{P}$ **do**
  Let $r_i.left$ equal $r_j \in M.base\_refs$ such that P_i is a subset of $P_j \in M.base\_sets$;
  Let $r_i.right$ equal $r_j \in N.base\_refs$ such that P_i is a subset of $P_j \in N.base\_sets$;
end for

The pseudocode Unknown_Interval_Lattice describes the final version of the unknown range algorithm optimized for both speed (using dynamic programming) and memory usage (using the set lattice data-type).

Unknown_Interval_Lattice$(\{G_t\}_{t=1}^{T})$
//**Input:** Graphs $\{G_t\}_{t=1}^{T}$.
//**Output:** A set lattice L containing each $\mathcal{P}(i, j)$.
Use DFS to obtain the connected components $\mathcal{P}(t, t)$ for every G_t.
$L_t.base\_sets = \mathcal{P}(t, t)$ for each t.
Initialize each of $L_t.base\_refs$ to a list of $|\mathcal{P}(t, t)|$ references with all $left$ and $right$ fields set to NIL.
for $k = 1$ to $T - 1$ **do**
  for $i = 1$ to $T - k$ **do**
    $\mathcal{P}(i, i+k) = \emptyset$;
    Compute the intersection $\mathcal{C}$ of $\mathcal{P}(i, i+k-1)$ and $\mathcal{P}(i+1, i+k)$.
    for $C_j \in \mathcal{C}$ **do**
      if $C_j \in \mathcal{P}(i, i+k-1)$ and $C_j \in \mathcal{P}(i+1, i+k)$ **then**
        $\mathcal{P}(i, i+k) = \mathcal{P}(i, i+k) \cup \{C_j\}$
      else
        $\mathcal{P}(i, i+k) = \mathcal{P}(i, i+k) \cup$ Int_Persistent$(\{G_t\}_{t=i}^{i+k}, C_j)$;
      end if
    end for
    $L =$ Combine$(L_i, L_{i+1}, \mathcal{P}(i, i+k))$;
    delete L_i; $L_i = L$;
  end for
  delete L_{T-k+1};
end for

23: **return** L_1;

A loop invariant at the end of each iteration of the inner loop is that L_i is the set lattice with the partition $\mathcal{P}(i, i+k)$ at its base. Thus, when $k = T-1$ and $i = 1$, L_1 will have the partition $\mathcal{P}(1, T)$ at its base, which is the desired set lattice.

6.3 Queries

After the set lattice is built, it is ready to be queried for more specific results. The following queries are examples of questions which may be answered quickly.

Query 1. Given i, j, what hidden groups are in the range $[i, j]$?

This query simply asks for the partition $\mathcal{P}(i, j)$, which contains the hidden groups which are persistent in the range $[i, j]$. A partition for any interval is built by starting at the base partition and following left and/or right references, combining sets which refer to the same coarser set in a subinterval. The algorithm In_Interval describes the general procedure for extracting the hidden groups in a particular range.

1: In_Interval(L, i, j)
2: //**Input:** Set lattice L, integers i, j.
3: //**Output:** A partition $\mathcal{P}$.
4: $\mathcal{P} = \{P_k\}_{k=1}^{N} = L.base\_sets$;
5: $R = \{r_k\}_{k=1}^{N} = L.base\_references$;
6: **for** $y = T$ downto j **do**
7: Let R' be the set $\{r' | r' = r_k.left\}$ with duplicates removed;
8: $S_{r'} \leftarrow \cup_{r_k.left=r'} P_k$ for each $r' \in R'$;
9: $\mathcal{P} \leftarrow \{S_{r'} | r' \in R'\}$; $R \leftarrow R'$;
10: **end for**
11: **for** $x = 1$ to i **do**
12: Let R' be the set $\{r' | r' = r_k.right\}$ with duplicates removed;
13: $S_{r'} \leftarrow \cup_{r_k.right=r'} P_k$ for each $r' \in R'$;
14: $\mathcal{P} \leftarrow \{S_{r'} | r' \in R'\}$; $R \leftarrow R'$;
15: **end for**
16: **return** $\mathcal{P}$;

Query 2. Given a duration t and a size h, what hidden groups are persistent for at least t cycles and contain at least h members?

Query 3. Given t, h and an actor or set of actors A, what hidden groups contain A, have at least h members, and last at least t intervals?

7 Extending the Model

We have assumed that the hidden group communicates, via a connected subgraph, in a contiguous subinterval of communication cycles. One could modify the algorithm to detect hidden groups that plan for at least C communication cycles, without requiring the cycles to be contiguous. Since we do not know *a priori* what the communication frequency of the hidden group is, a generalization that can lead to better results is obtained by relaxing the requirement that the *entire* hidden group is connected in *adjacent* communication cycles. However, the hidden group should still be connected "more often" than a randomly selected background group. More specifically, let S be a group of vertices. For $\epsilon \leq 1$, S is ϵ-*partially connected* if some subset of S of size at least $\epsilon \cdot |S|$ is internally connected. Partial connectivity quantifies the notion that only some fraction of the hidden group may be participating in the planning during a given communication cycle. We can now extend the notion of the hidden group so that it is partially connected in many communication cycles, possibly scattered over the entire observation range. We are thus led to the following more general problem definition, which we formulate here as a decision problem:

Problem:	FREQUENTLY_MOSTLY_CONNECTED (FMC)
Input:	A sequence of graphs $\{G_i(V, E_i)\}_{i=1}^{T}$ on the same vertex set V; positive integers $s \leq \|V\|$ and $k \leq T$, and the fraction ϵ, $0 < \epsilon \leq 1$;
Question:	Is there a subset $S \subseteq V$ with $\|S\| \geq s$ which induces an ϵ-partially connected subgraph in at least k of the $\{G_i\}$?

Our original formulation of the problem is a restriction of FMC to the case where $\epsilon = 1$ and $k = T$. In this case, we were able to give an efficient algorithm to solve the problem. For the more general problem, strong evidence exists to indicate that there is no such efficient algorithm:

Theorem 5 *Problem* FMC *is* NP*-complete.*

Proof: We show that a restriction of FMC is NP-complete. Suppose each graph G_i from our input-sequence is a clique $C_i \subseteq V$, and a set $V - C_i$ of isolated vertices. We set $\epsilon = 1$. We assume that $s > 1$. Then, it is easy to see that a set S is connected in a graph G_i *iff* it is a subset of C_i ($i \in [1, T]$). Thus, this restriction of FMC is equivalent to the problem of determining whether there is a subset of size s that is common to at least k of the sets $\{C_i\}$. This latter problem is known to be NP-complete ([10, page 112]). Thus we have a reduction from an NP-complete problem to a restriction of FMC, proving that FMC is also NP-complete. ■

The NP-completeness of FMC suggests that it is more prudent to search for a good approximate algorithm, or heuristic, to solve the problem. In the absence of any additional

information, one may employ a variety of approaches. However, it is more likely, that in any practical situation some additional information is available, for example, it may be known that certain (flagged) actors are part of the hidden group. In this case, one seeks to design an algorithm which finds only those hidden groups that contain the flagged subset of the vertices.

Further assumptions about the hidden group may significantly reduce the search. It is reasonable to assume that some members of such a group do not vary their communications within the group. A repeatedly occurring link highlights the actors involved in such communications. Another reasonable assumption is to limit the global pattern of communications within the hidden group. In our current analysis, we make only the assumption that the hidden group is connected. However, it is likely that these communications are represented by a sparse graph (*i.e.* very nearly a tree), and further, that the topology of this tree is changing very slowly with time.

8 Conclusion

Summary of Results. We introduced the notion of internally and externally connected subsets as a way to characterize the communication pattern of a hidden group. We then gave efficient algorithms to detect such hidden groups. Our simulations using random graph models of society communications indicates that if the background society communications are dense, then it is harder to detect the hidden group. In fact, a phase transition occurs when the average vertex degree exceeds $\ln n$, at which point hidden groups of moderate size become hard to detect. If the hidden group is trusting, or the background society communications are structured, the hidden group is also harder to detect.

Discussion. The experiments run in this study show that it is possible to identify persistently connected groups (which may be malicious) from structural properties of the communication graph, without using the contents of communications which may be misleading or difficult to decipher. Group identification done by an algorithm, such as the one proposed in this paper, could be used as the initial step in narrowing down the vast communication network to a smaller set of groups of actors. The communications among these potential hidden groups could then be scrutinized more closely.

Our results indicate a phase transition at $\ln n$ for the average number of communications that an actor makes in one communication cycle. For moderately sized societies, from 10,000 to 100,000 actors, this phase transition occurs at about 10; *i.e.*, if the average number of communications is 10 per actor per communication cycle, then it becomes hard to detect the hidden group. Thus, depending on the type of communication, the duration of the communication cycle may have to be made shorter to ensure that one is in the regime where a hidden group could be detected. Such an adjustment of the duration of a communication

cycle may not match the time scale which the hidden group ordinarily takes for a complete information exchange among its members. However closer to the planned event, they may need to communicate much more frequently, at which point they can be detected.

While the background communication density seems to be the dominant factor affecting our ability to detect the hidden group, we also find that if the society already has some structure (as with the group model), then it is harder to detect the hidden group. This result seems somewhat intuitive. However, a surprising result is that if the hidden group tries to hide all important communications within itself (a non-trusting group), it is *more easily* detected. The surprise becomes more obvious when the non-trusting groups are thought of as having more constrained communication patterns, and thus are able to be discovered with an appropriate algorithm.

A major assumption of the algorithms is that the time period Δ is known. There are numerous options for determining the correct value. One is through an intelligence source which may be able to, through experience with hidden groups, give an initial prediction of how often the group is expected to communicate. Another possibility is to run the algorithm many times with different values for Δ, examining the quality of the results obtained and changing Δ as appropriate. For example, if a given value of Δ yields no groups, Δ could be increased until substantial results are obtained. A final method is to use an adaptive window, where Δ changes in each interval so as to maximize the chances of discovery. Since the threshold density of edges for group discovery is $n \ln n$, the next interval could be determined by gathering the next $n \ln n$ communications from the stream of data.

There is are obvious tradeoffs between the time interval Δ, the number of intervals T, and the intensity of the background communications p. Δ needs to be large enough to ensure that all members of a hidden group communicate within that time interval. However, Δ must be small enough to ensure that the background p is not too dense ($O(\ln n/n)$) and that T is large enough (i.e. there are enough communication cycles to reliably detect hidden groups).

The algorithms presented extract both malicious and benign hidden groups (i.e. all groups that are planning in an organized way) from a large set of individuals. Given a set of n actors, there are potentially 2^n possible hidden groups. If every actor is in at most K planning groups, then there are $O(Kn)$ planning groups in total. Our algorithms would identify all these groups, but this is a significant improvement from having to monitor all the 2^n possible hidden groups, which would substantially decrease the amount of group monitoring and analysis needed. In the paper we present connectivity as one criterion for distinguishing a malicious group from all other groups. This criterion yield false positives. The goal of future research is to pare down the false positives by adding additional criteria, and eliminating false negatives through relaxation of criteria where appropriate. The algorithms presented here do not find hidden groups only, that is, they do not find only groups whose intention is to hide in a communication network, but are a step toward achieving that

goal. One way to minimize the number of false positives is to pre-filter the communications in order to eliminate certain messages that may be assumed to be non-malicious (such as broadcast messages with a large number of recipients).

Trusting and non-trusting hidden groups are a naming convention used to describe externally and internally connected hidden groups. The paper does not regard one or the other as the correct model for malicious groups; indeed, there may be some groups that use temporary "couriers" to hide communication paths, and there may be other groups which only use established routes. The hopeful argument is that malicious groups have to choose one of two options, where trusting is more risky in that secrecy could be compromised, while being non-trusting induces a structure that is more easily detected by our algorithms. Determining which behavior a particular malicious group or malicious groups in general exhibit could be the topic of further research.

The value $T(h)$ that we compute in our simulations, is actually an upper bound on the time to hidden group discovery. We assume that the hidden group is clever enough to hide among the very "heart" of the communication network, the part that stays connected longest. If instead, the hidden group is extracted from the large component earlier, a simple extension of our algorithm would find the group much more quickly. Also note that this analysis uses no semantic information whatsoever. If there is any additional information available, such as certain individuals who should be watched, or certain type of messages, our algorithms can be modified to yield a more efficient procedure for identification hidden groups.

Our results are of course only as accurate as our model is valid. However, we expect that the qualitative conclusions are robust with respect to the model. The extension of this work will explore more robust and realistic models of communication networks. Such models may additionally explore the notion of a *conversation* between two actors, where their communication tends to persist over a certain length of time instead of being random at every time period. Also, the groups can be made dynamic, where actors sometimes decide to enter or leave groups depending their preference.

Finally, we relaxed the assumption that the hidden group plans continuously during a given time period, or that every member of the hidden group participates in the planning during every communication cycle. In this general setting, we showed that there is strong evidence that the problem is not efficiently solvable, and one should resort to heuristics and hints (such as likely hidden group members).

References

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

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

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

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

[5] 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.

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

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

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

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

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, Ney York, 1979.

[11] 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 *Inaugural conference of the North American Association for Computational Social and Organizational Science (NAACSOS 2003)*, Pittsburgh, PA, June 2003.

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

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

[14] Malik Magdon-Ismail, Mark Goldberg, William Wallace, and David Siebecker. Locating hidden groups in communication networks using Hidden Markov Models. In *International Conference on Intelligence and Security Informatics (ISI 2003)*, Tucson, AZ, June 2003.

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

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

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

[18] 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.

[19] 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.

[20] Thomas A. Stewart. Six degrees of Mohamed Atta. *Business 2.0*, 2 issue 10:63, 2001.

[21] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, U.S.A., 2001.