DIMACS Technical Report 2005-28
September 2005

A Framework for Analysis of Dynamic Social Networks

by

Tanya Y. Berger-Wolf[1]
Center for Discrete Mathematics and Theoretical Computer Science (DIMACS)
CoRE Bldg, Rutgers University
96 Frelinghuysen Rd., Piscataway, NJ 08854
and
Department of Computer Science, University of Illinois at Chicago,
851 S. Morgan, Chicago, IL 60607
tanyabw@cs.uic.edu

Jared Saia[2]
Department of Computer Science, University of New Mexico,
Albuquerque, NM 87131
saia@cs.unm.edu

[1]DIMACS Postdoctoral Fellow, suported by the National Science Foundation postdoctoral fellowship grant EIA 02-05116
[2]Supported by the National Science Foundation grant CCR-0313160 and by the Sandia University Research Program Grant No. 191445

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research, Microsoft Research, Stevens Institute of Technology, Georgia Institute of Technology and Rensselaer Polytechnic Institute. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

Finding patterns of social interaction within a population has wide-ranging applications including: disease modeling, cultural and information transmission, phylogeography, conservation, and behavioral ecology. Recently, scientists have started to model social interaction with graphs (networks). One of the intrinsic characteristics of societies is their continual change. However, the majority of the social network analysis methodologies today are essentially static in that all information about the time that social interactions take place is discarded or long time series are averaged to discern the overall or long-term strength of connections. Such approach may not only give inaccurate or inexact information about the patterns in the data, but it prevents us from even asking questions about the temporal causes and consequences of social structures. In this paper we propose a new mathematical and computational framework that allows analysis of dynamic social networks addressing the time component explicitly. We present several algorithms that explore the social structure in this model and pose many open questions.

1 Introduction

Finding patterns of social interaction within a population has wide-ranging applications including: disease modeling [11, 15, 21], cultural and information transmission [3, 5, 6, 8, 16, 31, 32, 35], intelligence and surveillance [2, 3, 20, 26, 25], business management [4, 7, 29, 30], conservation biology and behavioral ecology [9, 10, 23]. Until recently, questions regarding social structures and social interactions in human or animal populations were answered using statistical tools, which summarized aggregate information over the population. Recently, scientists have started to model social interaction with graphs (networks) [13, 17, 18, 19, 27, 28, 24, 33, 34].

Typically, each individual is represented by a node in the graph, and there is an edge between two nodes if a social interaction occurred at any point in time between the two individuals represented by these nodes. Depending on the source of data, a social interaction could be a verbal or written communication, scientific collaboration, sexual contact, physical proximity, affiliative or antagonistic contact, or potential disease transmission. Sometimes, the edges in the graph are weighted with the frequency or the probability of interaction.

The technique of modeling social interactions using this type of graph has been very successful. However, a major drawback of this model is that it is essentially static in that all information about the time that social interactions take place is discarded or long time series are averaged to discern the overall or long-term strength of connections. The static nature of the graph gives rise to two major problems. First, it can give us inaccurate or inexact information about patterns in the data. For example, as Figure 1 illustrates, in some cases, very different dynamic data can give rise to the same static graph. The decisions made based solely on the static data may be flawed. For example, in the epidemiological setting, Figure 1 represents potential disease transmission edges. Suppose we are interested in vaccinating a limited number of people (1 person for the purpose of this example) so as to minimize the total number of infected individuals at the end, assuming any contact will facilitate the transmission and any individual may be infected initially. In the static graph there is no difference between the three individuals and it seems no matter which one individual is vaccinated there will be two people infected at the end. However, the dynamic situation may be different and other factors may come into play. In the first case, assuming the initial infection starts at step 0 and lasts for no longer than 2 time steps, only b would need to be vaccinated, and no matter whether a or c is infected initially, only one individual will stay infected.

A second, more serious, problem with the static graph representation is that it prevents us from even asking certain fundamental questions about either the causes or consequences of social patterns. How quickly can a disease spread through the population? Which individuals would we inoculate to slow down its spread? How does the size and stability of social structures change with outside circumstances (e.g. season, time of day, predator activity, upcoming conference or journal deadlines, court subpoenas, terrorist activities)? What is the average lifespan of a social structure and are most structures long-lived or short-lived? Are there differences in the life span of social structures with respect to their size and the demographics of their members? To be able to answer these questions, we need to have

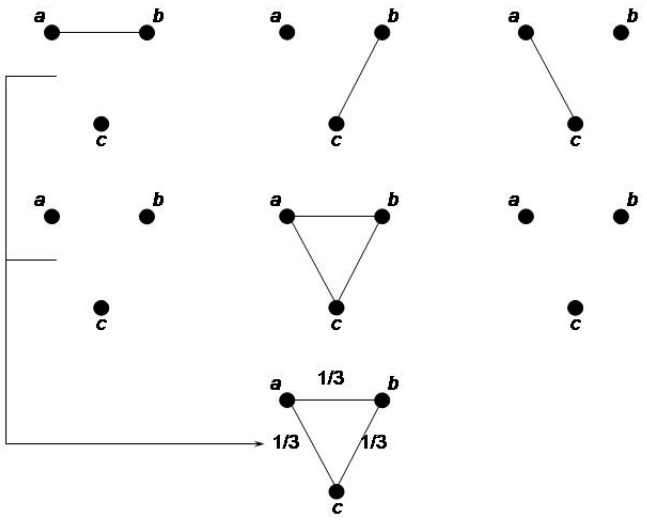

Figure 1. The top two rows represent social interactions over three time steps. The bottom row represents the static graph obtained. The social interactions specified in the first row of graphs are very different from the social interactions specified in the second row of graphs. However both sets of data give rise to the same static graph.

information on when social interactions occurred.

To address the dynamic aspect of social interactions we use the graph (network) population model to capture the time snapshot information. We then build a time series graph that connects the snapshot information. The connections within a time snapshot are the individual associations that exist at that moment. We impose a metric on sets of individuals within different time steps that allows us to track changes in social structures over time. In this model, many questions about the social population dynamics become the classical questions of graph connectivity. Some of these are easily answered, yet many pose challenging algorithmic problems. We now describe this approach and our assumptions in more detail.

The rest of the paper is organized as follows: in Section 2 we discuss the type of input our approach assumes and formally state the dynamic social network analysis framework. In Section 3 we present algorithms for computing some basic properties of the dynamic social network. We give algorithms for finding the most persistent and the largest social structures, as well as social structures that encompass a set of specified groups of individuals. Section 4 lists a collection of open problems with the related population questions and comments on some possible solution approaches.

2 The Formal Dynamic Social Networks Analysis Framework

Here we present our mathematical and computational framework, the context, the assumptions and the necessary definitions. We start by discussing the input data.

2.1 The Input Data

To be able to answer questions about social structure in a dynamic setting, we assume that the input is as follows. A population of individuals is monitored in some way. There are three types of input data:

1. Information about each individual such as id number and GPS or other type of location or action identification. Pairwise connection data has to be inferred (from the geometric proximity data of GPS or from a common part of the action - in surveillance data).

2. Pairwise connection data (from cellphone communication, emails, sexual contact information, co-publication, etc.). Group information has to be inferred.

3. Direct group information, such photographs, satellite data, class registration roster, etc. These may provide either an instant complete snapshot of the entire population (satellite data) or a time window snapshot of some parts of the population.

For the purpose of this paper, we assume that the input is in the form of the partition of the individuals into groups at every time step. This may be a result of the direct observation data or the output of processing of the pairwise connection data.

2.2 Definitions

We now state the formal input representation. Given a population $X = \{x_1, ..., x_n\}$, we define a *group* to be a subset $g \subseteq X$. We assume that the input is a set of partitions, $P_1, P_2, ..., P_T$, and that each partition, P_i, is a set of disjoint *groups*:

$$\forall i, \quad 1 \leq i \leq t \quad P_i = \{g_{i1}, ..., g_{ii_m}\}, \quad \cup_{j=1}^{i_m} g_{ij} = X \text{ and}$$

$$\forall p \neq q, \quad 1 \leq p, q \leq i_m \quad g_{ip} \cap g_{iq} = \emptyset$$

We denote by $P(g)$ the index of the partition to which g belongs. That is, if $g \in P_i$ then $P(g) = i$.

Given two groups, g and h, a set similarity measure $sim(,)$, and a *turnover threshold* β, we consider the two groups *similar* if

$$sim(g, h) \geq \beta.$$

Our definition here is independent of the set similarity measure. There may be many possible such measures, from the standard Jaccard similarity measure [14] (the size of the intersection

over the union) to application-specific measures, such as genomic distance, that may be NP-hard to compute. There are, however, a few universal properties of the similarity measure we assume:

1. $sim(h, g)$ depends only on h and g.
2. $sim(g, g)$ = maximum similarity value.
3. $sim(h, g)$ monotonically increases with the increase of $|h \cap g|$ when $|h| + |g|$ does not change.
4. $sim(h, g)$ monotonically increases with the decrease of $|h| + |g|$ when $|h \cap g$ does not change.

For the purposes of this paper, we also assume that for any two groups we are given their set similarity measure, either by easy (polynomial) computation or from an outside source.

We now define the main concept of our framework - a metagroup.

Definition 1. *Given an input in the form of partitions $P_1, ..., P_T$ of a set of individuals X, a set similarity measure $sim(,)$, a turnover threshold β and a function $\alpha(T)$, a* metagroup *MG is a sequence of groups $MG = \langle g_1, ..., g_l \rangle$, $\alpha(T) \leq l \leq T$ such that*

1. *no two groups in MG are in the same partition and the groups are ordered by the partition time steps:*

$$\forall i, j, \quad 1 \leq i < j \leq l, \quad P(g_i) < P(g_j),$$

2. *the consecutive groups in MG are similar:*

$$\forall i, \quad 1 \leq i < l, \quad sim(g_i, g_{i+1}) \geq \beta.$$

We call the parameter α the persistence *of a metagroup.*

We note that the intersection between g_1 and g_l may be null by this definition; our only constraint is that the groups change gradually. A metagroup is a timeline evolution history of a group of individuals. Just like any real group, it may change its members over time (unless β is too high to allow it), yet locally remains essentially the same group. Our definition of a metagroup is flexible enough to accommodate the range of social structures, from the "never changing, always together" to the "meeting every now and then of more or less the same individuals".

Definition 2. *An individual $x \in X$ is a* member of a metagroup *$MG = \langle g_1, ..., g_l \rangle$ if the number of groups $g_1, ..., g_l$ to which x belongs is at least some a priori chosen* membership threshold *function γ (which may be constant or a function of T, the total number of individuals associated with MG, $\alpha(T)$, and the average number of times any individual appears in MG).*

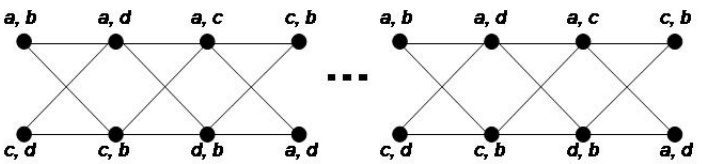

Figure 2. Here the turnover threshold β is such that groups sharing half of their members are considered similar. Each path from a group in P_1 to a group in P_T is a metagroup. Any path has a choice of two vertices in any time step, hence there are 2^T possible such paths while there are only $2T$ groups. Here $P_1 = \{\{a, b\}, \{c, d\}\}$, represented by the two leftmost vertices, $P_2 = \{\{a, d\}, \{c, b\}\}$, etc.

The precise values of α (persistence), β (turnover) and γ (membership), the meaning of a "group" and the similarity measure are application specific. Our goal is to create a framework that is independent of these definitions and is capable of providing meaningful answers for a wide range of applications.

We use a weighted multipartite directed graph for the conceptual representation: $G = (V_1, ..., V_T, E)$ where V_i is the set of groups in partition P_i and $(g_i, g_j) \in E$ if $P(g_i) < P(g_j)$ and $sim(g_i, g_j) \geq \beta$. Note that this is a directed acyclic graph (DAG) since all the edges are directed from an earlier time step to a later one. The weight $w(g_i, g_j) = sim(g_i, g_j)$. A metagroup in this graph is a path of length at least $\alpha(T)$. We shall call this graph a *metagroup β-graph*.

Note that the number of metagroups may easily be exponential in the number of time steps T even if the total number of groups is small and $\alpha(T) = T$. Figure 2 shows an example of such a graph. Thus, it may be impractical to explicitly list all existing metagroups. However, it is also unnecessary to do so to answer many important and interesting questions, such as the most persistent or the largest metagroup, the total number of metagroups and the connectivity of a given subset of groups. We now present a list of algorithms to answer specific queries about the metagroups.

From now on we assume the definition of a metagroup that satisfies a priori given thresholds of α and β. We assume the input is in the form of a metagroup β-graph described above with the present edges of weight no less than β.

3 Problems and Solutions

We now show how questions about social structures and interactions can be algorithmically answered in our framework. For all of the problems we assume the definition of a metagroup that satisfies a priori given thresholds of α and β and an individual membership that satisfies a threshold γ. We assume the input is in the form of a metagroup β-graph described above with the present edges of weight no less than β.

3.1 Metagroup Statistics

While it might be impractical to list all the metagroups in some cases, there are nonetheless many statistics we can calculate efficiently, such as the number of the metagroups and the average metagroup lifespan. The number of metagroups is the number of paths of length at least α in the graph and the average metagroup length is the average length of a path which is at least α long. Both can be computed with the same simple dynamic programming algorithm we present below. Let $P(g, l)$ denote the number of paths of length exactly l from a minimal vertex to a group g. We compute the table $P(g, l)$ starting with the groups in partition 0 and moving forward in time to partition T.

Alg METAGROUPS STATISTICS

$$\forall g, 0 \leq l \leq P(g) \quad P(g,l) = \begin{cases} 1 & \text{if } g \text{ is minimal and } l = 0 \\ 0 & \text{if } g \text{ is minimal and } l > 0 \\ \sum_{(h,g) \in E} P(h, l-1) & \text{otherwise} \end{cases}$$

Total number of metagroups

$$N(MG) = \sum_{\substack{\text{maximal } g \\ l \geq \alpha}} P(g,l)$$

Average metagroup length

$$AL(MG) = \frac{\sum_{\substack{\text{maximal } g \\ l \geq \alpha}} P(g,l) \times l}{N(MG)}$$

Maximal metagroup length

$$MaxL(MG) = \max_{\substack{\text{maximal } g \\ l \geq \alpha, P(g,l) > 0}} \{l\}$$

Minimal metagroup length

$$MinL(MG) = \min_{\substack{\text{maximal } g \\ l \geq \alpha, P(g,l) > 0}} \{l\}$$

3.2 Most Persistent Metagroup

One of the fundamental questions in the dynamic analysis of social connections is that of their persistence [1]. That is, are the connections fleeting, a chance meeting or a consistent long term relationship among the members of a group. We formulate this question computationally as finding the longest metagroup. MAXIMUM PERSISTENCE METAGROUP: Find a metagroup MG which maximizes the number of groups associated with MG.
Solution: The question is equivalent to finding the longest path in a DAG, which is a well studied problem and can be solved in linear time using dynamic programming on a topologically sorted graph.

3.3 Largest Metagroup

Here we are interested in the metagroup with the largest mebmership. Thus, we assume that for each group node we know which individuals are in that gorup. LARGEST METAGROUP: Find the metagroup MG which maximizes the number of members of MG.

Solution: We use dynamic programming to solve this problem. Let $S(g,k,l)$ = the *set* of individuals that appear at least k times in a metagroup of length at least l so far that ends at the current group g and $N(g,k,l)$ keep track of the incoming edge of the metagroup to which g is assigned. Recall that $P(g)$ is the index of the partition to which g belongs.

Alg LARGEST METAGROUP

$\forall g,$ s.t. $P(g)=0,$ $\quad S(g,k,l) = \begin{cases} g & \text{if } k=1 \text{ and } l=0 \\ \emptyset & \text{otherwise} \end{cases}$

$\forall g,$ $\quad S(g,0,0) = X$

$\forall g, \forall l > 0,$ $\quad S(g,0,l) = S(h_{\max}, 0, l-1)$
where $|S(h_{\max},0,l-1)| = \max_{(h,g)\in E} |S(h,0,l-1)|$
$N(g,0,l) = h_{\max}$

$\forall g, \forall l > 0, k > 0,$ $\quad S(g,k,l) = (S(h_{\max},k,l-1) - g) \cup (S(h_{\max},k-1,l-1) \cap g)$
where $h_{\max}$ achieves
$\max_{(h,g)\in E} |S(h,k,l-1) - g| + |S(h,k-1,l-1) \cap g|$
$N(g,k,l) = h_{\max}$

Let g_{max} be a group such that

$$|S(g_{max},\gamma,\alpha)| = \max_g |S(g,\gamma,\alpha)|$$

Trace back the corresponding metagroup MG using the $N(g_{max},\gamma,\alpha)$ entry as a starting point.
Return MG.

First, we address the correctness of the LARGEST METAGROUP algorithm. Consider any metagroup. It must contain at least α groups and its members are the individuals that appear in at least γ of those groups. Thus, for the last group g of the metagroup, $S(g,\gamma,\alpha)$ tracks precisely the set of its members. We now show that we correctly compute this set.

- Any minimal group can only be the endpoint of metagroups of length 0 and the groups members are the individuals that appear there exactly once. No other option is possible for a minimal group.
- Trivially, any group is the endpoint of a zero-length metagroup (itself) and every individual appears in it at least zero times.
- What are the individuals that appear in a metagroup of length l at least 0 times? If the current group has a path of length at least l to it from a minimal vertex, then,

of course, it's the entire population, otherwise there is no such metagroup (and its membership is an empty set). In other words, if g has an incoming edge from a group h such that h is an endpoint of a path of length at least $l-1$ from a minimal vertex then the members of g are X, otherwise it's the empty set. This is exactly the first recursive statement of the algorithm.

- Finally, the individuals that appear at least k times in a metagroup of length at least l that ends at g may either appear in g or not. In the former case they must already appear at least k times in a metagroup of length at least $l-1$ that ends at one of the incoming neighbors of g. In the latter case they must appear at least $k-1$ times by the time they get to g. This is precisely the general recursive statement of the algorithm.

Thus, the algorithm correctly computes the set of individuals that appear at least γ times in a metagroup of length at least α that end at g for every vertex g, which is precisely the membership of a metagroup that ends at g. The algorithm thus correctly returns the metagroup with the largest such membership set.

We now discuss the running time of the Largest Metagroup algorithm. Clearly, no individual can be in a metagroup more times than the length of the group plus one (due to the initial time step being 0). Therefore no matter how complicated the function γ may be it is always no greater than T and so is, of course, α. Thus, the size of the $S(g,k,l)$ table is $O(mT^2)$, where m is the number of groups. To compute an entry in this table we need to perform two intersections with g and a union of the two resulting sets, which takes $O(n)$ time in the worst case if the individuals in each set are sorted (recall that $n = |X|$ is the number of individuals). These operations have to be done for each neighbor of g to find the maximum of which there may be no more than m. Thus, the total worst case running time of the algorithm is $O(m^2T^2n)$ or $O(mT^2n|E|)$ since each edge in the graph is considered exactly once.

3.4 Group Connectivity

Another natural question to ask is, given a collection of groups at different time steps, whether they belong to the same overall social structure, i.e. the same metagroup. Formally, there are several computational problems of Group Connectivity: Let $g_1, ..., g_l$ be a set of groups in separate partitions, sorted by the indices of the partitions in which they appear (i.e., $P(g_i) < P(g_{i+1})$).

- Is there a metagroup that contains all the groups?
- Find the most persistent metagroup that contains all groups.
- Find the largest metagroup that contains all groups.
- Find the metagroup that contains the maximum number of groups.

Figure 3. A *beta*-component example. Every vertex g_i is a cut vertex and all the minimal vertices are in partitions smaller than $P(g_1)$ and the maximal vertices are in partitions greater than $P(g_l)$.

To answer all of these questions we first preprocess the graph in the following way. The goal is to find a connected component with the vertices $g_1, ..., g_l$ being the cut vertices and all the minimal vertices being in partitions before g_1 (or g_1 itself) and all maximal vertices being in partitions after g_l (or g_l itself). Figure 3 illustrates the general structure of such a subgraph. We shall call this subgraph a *β-component* of $g_1, ..., g_l$. The algorithm Beta-Component computes the β-component of the vertices $g_1, ..., g_l$. Here BFS stands for the Breadth-First Search.

Alg Beta-Component:

FOR $1 \leq i \leq l-1$ **DO**
BFS from g_i to g_{i+1}, marking each encountered vertex.
Reverse BFS from g_{i+1} to g_i retaining only the vertices that have been previously marked by g_i.
Add the vertices traced by the reverse BFS from g_1 and the forward BFS from g_l.
RETURN the subgraph induced by the selected vertices.

Proposition 1. *Algorithm* Beta-Connected *returns a subgraph in which every vertex g_i is either a cut vertex, a minimal vertex, or a maximal vertex and there are no minimal or maximal vertices that are not one of the g_i's in partitions greater than g_1 or less than g_l. Moreover, if the returned subgraph is connected then every path from a minimal to a maximal vertex in it contains all the vertices $g_1, ..., g_l$.*

Proof. **Each g_i is either a cut, a minimal, or a maximal vertex.** Suppose there exists a g_i which is not a cut vertex and let it be such a vertex with the smallest i. If it

is neither a minimal or a maximal vertex then there is a path from a minimal vertex $g_{\min}$ (with $P(g_{\min}) < P(g_i)$) to a maximal vertex $g_{\max}$ (with $P(g_{\max}) > P(g_i)$) that does not pass through g_i. Let this path be $g_{\min} = v_0, v_1, ..., v_p = g_{\max}$. Every edge (v_j, v_{j+1}) on this path must be traced either by a reverse BFS or a forward BFS from g_i (otherwise there is an edge that has not been marked by g_i between g_{i-1} and g_{i+1}). Let (v_{j-1}, v_j) be the last edge which was traced by the the reverse BFS from g_i. Notice that no edge before that can be traced by the forward BFS from g_i since that would complete a cycle. However, the edge (v_j, v_{j+1}) cannot be traced by a forward BFS from g_i since that implies a path from g_i to v_j and since there is a path from v_j to g_i (for the reverse BFS) then that completes a cycle. Thus, every g_i is either a minimal, a maximal, or a cut vertex.

Every minimal vertex is either a g_i or is in a partition smaller than $P(g_1)$. Suppose v is a minimal vertex which is not a g_i and there is a j such that $P(g_j) < P(v)$. Since v is a minimal vertex which was retained in the resulting graph then there is a path from v to some g_i. For that path to exist every edge on that path has to be reached both by the reverse BFS from g_i and by the froward BFS from g_{i-1}. Unless $v = g_{i-1}$, v has an incoming edge and hence is not minimal.

Similarly, **every maximal vertex is either a g_i or is in a partition greater than $P(g_l)$.**

If the returned graph is connected, every path from a minimal to a maximal vertex in it contains all the vertices $g_1, ..., g_l$. Suppose, to the contrary, the returned subgraph is connected, but there is a minimal to maximal vertex path that does not contain g_i. That means that either g_i is not a cut vertex or (similar to the above cut vertex proof) there is an edge that has not been traced both by g_i and g_{i-1}. □

With this β-component we are ready to answer the various questions we have listed about group connectivity.

- Is there a metagroup that contains all the groups?
  Alg: If the β-component of the groups is connected then the question is equivalent to whether there is a path from a minimal to a maximal vertex of length at least α.

- Find the most persistent metagroup that contains all groups.
  Alg: This is equivalent to finding the longest minimal-maximal vertex path in a connected β-component. We return such a path if it is of length at least α.

- Find the largest metagroup that contains all groups.
  Alg: Use the LARGEST METAGROUP algorithm on the β-component to find the largest metagroup that contains all groups.

- Find the metagroup that contains the maximum number of groups.

Alg:
Dynamic Programming: Let $M(g)$ be the maximum number of groups from among $\{g_1, ..., g_l\}$ in a metagroup that ends at g.

$$M(g_1) = 1$$

$$\forall g, \text{ s.t. } P(g) \leq P(g_1), \quad M(g) = 0$$

$$\forall g, \text{ s.t. } P(g) > P(g_1), \quad M(g) = \begin{cases} \max\limits_{(h,g)\in E} \{M(h) + 1\} & \text{if } g \in \{g_1, ..., g_l\} \\ \max\limits_{(h,g)\in E} \{M(h)\} & \text{otherwise} \end{cases}$$

Return $\max_{1\leq i\leq l} \{M(g_i)\}$. As in any dynamic programming algorithm, backtrace the actual metagroup that corresponds to that number.

3.5 Individual Connectivity

Similar to group connectivity, we may ask questions about the individual connectivity. That is, given set of individuals, whether they belong to the same social structure. More formally, INDIVIDUAL CONNECTIVITY: Let $S \subseteq X$ be a set of individuals. We assume that for each group we have the corresponding list of individuals that comprise it. We change the order of questions slightly from the group connectivity list.

- Find the metagroup that contains as many as possible of the individuals in S as members.
  This is equivalent to the LARGEST METAGROUP problem with the set of individuals X limited to S on the original metagroup β-graph. That is, let the underlying graph be the original metagroup β-graph. For each group let the set of individuals associated with that group be the original set intersected with S. Now perform the LARGEST METAGROUP algorithm on this new graph.

- Is there a metagroup that contains all the individuals in S as its members?
  Certainly, if the metagroup containing the largest number of individuals from S (see previous) contains them all then the answer to this question is "yes", otherwise, it is "no". Yet, there may be a more efficient way of answering the existence question alone. We leave it as an open question.

- Find the most persistent metagroup that contains all the individuals in S as its members.
  Again, we use the LARGEST METAGROUP algorithm with slight modifications. We restrict X to S, as above and vary l from 1 to T rather than to $\alpha(T)$. (Recall that the table $S(g, k, l)$ tracks the set of individuals that appear at least k times in a largest metagroup ending g of length at least l and $N(g, k, l)$ records the number of such individuals.) As we have pointed out, the vertices g with $N(g, \gamma, l) = |S|$, where $l \geq \alpha$, are the end points of metagroups that contain all the individuals in S. Among all such

vertices we choose g_{end} with the largest l and trace the actual metagroup backwards through the dynamic programming table to find the most persistent metagroup that contains all the individuals in S.

- Find the largest metagroup that contains all the individuals in S as members.
  First, we find the graph of the metagroups that contain all the individuals in S using the LARGEST METAGROUP algorithm with the set X restricted to S. As before, the vertices g with $N(g, \gamma, l) = |S|$, where $l \geq \alpha$, are the end points of the metagroups that contain all the individuals in S. We can trace back the subgraph H (not necessarily induced) where every path of length at least α is a metagroup that contains S. On that subgraph we can now run the LARGEST METAGROUP algorithm with the original X to find the largest metagroup that contains S.

  Proposition 2. *The algorithm described above returns the largest metagroup that contains the individuals in S.*

3.6 Critical Group Set

Until now, all the questions are the fundamental ones that may be asked about a dynamic social network and it is fortunate that we can answer them in polynomial time. However, it would be naive to expect that all the aspects of a dynamic social network can be explored so efficiently. For example, the question of fragility of a social network can be formulated in many ways. One way to ask it is to consider the smallest set of groups whose absence would leave no recognizable overall social structures. Formally, CRITICAL GROUPS SET: Find the smallest set of groups whose removal leaves no metagroups (with respect to given β and α). This question is particularly important in an epidemiological context where it is safe to have prolonged interactions for periods no longer than the non-contagious incubation period of a disease, yet it is important to quarantine people otherwise. In this context some groups may be considered more important (or easier) to dissipate than others and this would lead to a vertex-weighted version of the problem. More generally, MIN K-PATH VERTEX SHATTERING SET: for an arbitrary graph $G = (V, E)$ find the smallest (weighted) subset of vertices $U \subseteq V$ such that the subgraph induced by $V - U$ has no paths longer than $k - 1$. We show that the complexity of this problem very much depends on the specific value of k.

First, we show that the general problem is NP-complete by demonstrating that the unweighted MIN 2-PATH VERTEX SHATTERING SET is equivalent to MAX INDEPENDENT SET, which is a well studied NP-complete problem.

Theorem 1. MIN 2-PATH VERTEX SHATTERING SET *is polynomially equivalent to* MAX INDEPENDENT SET.

Proof. The MIN 2-PATH VERTEX SHATTERING SET problem is to remove the smallest subset of vertices so that the remaining induced subgraph has no paths, i.e. the remaining subgraph is an independent set. Since the set of removed vertices and the independent set

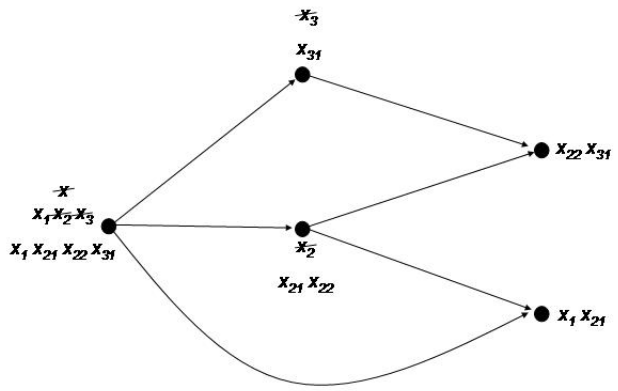

Figure 4. An example of an assignment of individuals to vertices in an arbitrary DAG.

do not intersect, finding the largest independent set is equivalent to finding the smallest complimentary set. □

Corollary 1. Min k-Path Vertex Shattering Set *on an arbitrary graph G is NP-complete.*

The special structure of a metagroup β-graph does not improve the complexity of the Min 2-path Vertex Shattering Set problem.

Theorem 2. Min 2-path Vertex Shattering Set *on a metagroup β-graph is NP-complete.*

Proof. Let $G = (V, E)$ be an arbitrary instance of Max Independent Set. Starting with any vertex v, we use the orientation induced by the breadth first search (BFS) from v to obtain a directed acyclic graph on G. To ensure that this graph is a metagroups β-graph we need to assign individuals to every vertex consistent with the edges. We do this in the following way: starting with the minimal (no incoming edges) vertex v, we assign it a set x and proceed in the BFS order. For every vertex u with the currently representative individual x_p, we assign to its q children the individuals $x_{p1}, x_{p2}, ..., x_{pq}$, respectively, and replace x_p by the set $\{x_{p1}, x_{p2}, ..., x_{pq}\}$ at every vertex it appears, all the way to the root. If one of the children had another individual y_i already assigned to it, we add the additional assignment induced by u. Figure 4 shows an example of such an assignment. For each vertex u we calculate the longest path distance from v in the induced DAG, $d(u)$. A vertex u now corresponds to a group in a time step $d(u)$. It is not difficult to see that each individual appears at most once in any vertex in a given time step. It is now left to ensure that each

individual appears exactly once in every time step. To do so, for every edge (u_1, u_2) such that $d(u_2) - d(u_1) > 1$ there exists an alternate path from u_1 to u_2 with every edge shorter than $d(u_2) - d(u_1)$. We add the individuals in u_2 to every vertex on that path. For example, in Figure 4 the individual x_1 will be added to the individuals x_{21}, x_{22} in time 1. Again, it is not hard to see that after this procedure for every edge longer than 1 every individual appears exactly once in every time step. Therefore, the set of vertices at every time step represents a partition of the individuals in the population. We use the similarity measure of the size of the intersection of the two sets, $sim(h, g) = |h \cap g|$ and set $\beta = .5$. With these parameters, the resulting graph is a metagroup β-graph. If we can solve the MIN 2-PATH VERTEX SHATTERING SET on a metagroup β-graph, we can use it to find a MAX INDEPENDENT SET in an arbitrary graph. Therefore, MIN 2-PATH VERTEX SHATTERING SET in a metagroup β-graph is NP-hard. Given a solution to the MIN 2-PATH VERTEX SHATTERING SET it is easy to verify its correctness, therefore the problem is NP-complete. □

The general problem of finding the minimum number of vertices to be deleted to obtain a subgraph with property P is well studied [22]. For any non-trivial hereditary property P the problem is APX-hard [22]. A graph property is hereditary if it is closed under taking induced subgraphs. In our case, the property $P =$ THE GRAPH HAS NO PATHS OF LENGTH k OR GREATER is certainly hereditary. However, it does not naturally fall into any of the known constant factor approximation requirements.

While the MIN k-PATH VERTEX SHATTERING SET problem is NP-hard for an arbitrary k, we present a polynomial time algorithms for $k = T$ in a DAG, where T is the length of the longest path in G. Let $P(v, l)$ be the number of paths of length exactly l from a minimal vertex to $v \in V$. The algorithm uses the following function K-GRAPH:

Alg K-GRAPH(k)
In the order of a BFS from the minimal vertices:

$$\forall v, l, \quad 0 \le l \le T \quad P(v,l) = \begin{cases} 1 & \text{if } v \text{ is minimal and } l = 0 \\ 0 & \text{if } v \text{ is minimal and } l > 0 \\ \sum_{(u,v)\in E} P(u, l-1) & \text{otherwise} \end{cases}$$

FOR all maximal v and $l \ge k$ s.t. $P(v, l) > 0$ **DO**
    Trace back the paths of length l that terminate at v
RETURN the resulting traced graph G_k

Lemma 1. *Algorithm k-GRAPH(k) returns a subgraph G_k of G that contains every edge in G which is in some path of length at least k from a minimal to a maximal vertex in G. That is, G_k is the subgraph induced by the set of edges that are in some path of length at least k from a minimal to a maximal vertex in G.*

Proof. Suppose, to the contrary, there exists an edge in G which is in some path of length at least k (denote k-path), $v_{\min} = v_0, ..., v_l = v_{\max}$, yet it does not appear in G_k. Let (v_{i-1}, v_i)

be an edge with the maximum value of i among all such edges. If v_i is a maximal vertex then (v_{i-1}, v_i) is the last edge on a k-path and v_i would be selected for a traceback by the algorithm. The algorithm selects all the vertices x such that $(x, v_i) \in E$ and $P(x, l) > 0$ for $l \geq k - 1$. Certainly v_{i-1} is such a vertex and would be selected. Suppose then v_i is not a maximal vertex. Since (v_{i-1}, v_i) is on a k-path then the edge (v_i, v_{i+1}) is also on a k-path. Since (v_{i-1}, v_i) was chosen to have the maximum i among all unselected edges then the edge (v_i, v_{i+1}) must be in G_k. By the same argument, every edge (v_j, v_{j+1}) for $j \geq i$ is in G_k. However, every edge in G_k is the result of a traceback from a maximal vertex, including $v_{\max}$. $P(v_{\max}, l) > 0$ and $l \geq k$, therefore the algorithm *will* trace back to v_i with the value $P(v_i, l - i)$. Since $P(v_i, l - i) > 0$, the trace will continue to all $(x, v_i) \in E$ such that $P(x, l - i - 1) > 0$. But v_{i-1} is one of such vertices x and the edge (v_{i-1}, v_i) will be selected by the traceback. This is a contradiction, therefore every edge on a k-path in G will be present in G_k. □

Corollary 2. *G_k contains all the paths of length at least k from a minimal to a maximal vertex in G.*

The running time of the K-GRAPH algorithm is $O(|E||V|)$ since each edge is traced exactly once and $T \leq |V|$. Using the function K-GRAPH we can state the MIN T-PATH VERTEX SHATTERING SET solution.

Alg MIN T-PATH VERTEX SHATTERING SET G_T = K-GRAPH(T)
Add a vertex s connected to all the minimal vertices and a vertex t connected from all the maximal vertices in G_T
U = minimal vertex cut in the resulting graph $G_T + \{s, t\}$
RETURN U.

Theorem 3. *Algorithm* MIN T-PATH VERTEX SHATTERING SET *is a polynomial time algorithm that returns the smallest subset of vertices U whose removal leaves a subgraph with no path of length T (where T is the length of the longest path in the original graph G).*

Proof. We prove the theorem using the following two lemmas.

Lemma 2. *G_T contains all and only paths of length T from a minimal to a maximal vertex in G.*

Proof. By Lemma 1, every edge in G_T is in some path of length T in G. Moreover, since T is the length of a longest path in G, every path from a minimal to a maximal vertex in G_T is of length T. Suppose, to the contrary, there is a path $v_{\min} = v_0, ..., v_l = v_{\max}$ which is shorter than T. For every edge (v_{i-1}, v_i) at least one of the following must hold:

1. there is a path of length $T - i$ from v_i to a maximal vertex, or
2. there is a path of length $T - l + i - 1$ from a minimal vertex to v_{i-1}.

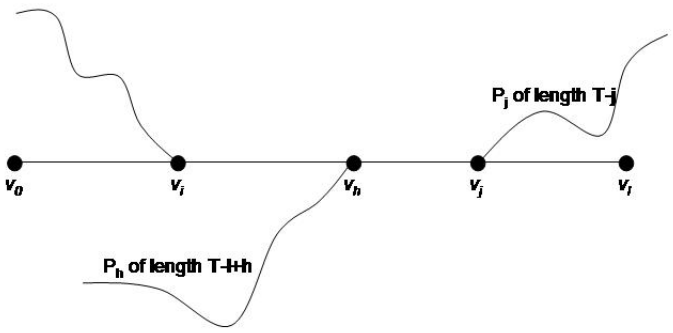

Figure 5. Illustration of the proof of Lemma 2.

Let v_i be the vertex with the smallest i such that its incoming degree in G_T is greater than 1. (There must be such a vertex, otherwise every vertex v_i, including v_l, has only one incoming path, namely the path from $v_{\min}$, and therefore every edge (v_{i-1}, v_i) is only on that unique path from $v_{\min}$ to $v_{\max}$ which is of length $l < T$.) Since there is no path of length $T-l+i-1$ from a minimal vertex to the vertex v_{i-1}, there must be a path of length $T-i$ from v_i to a maximal vertex. It cannot be a path $v_i, v_{i+1}, ..., v_l$ since that one is of length $l-i < T-i$. Thus, there must be a vertex v_j, $j \geq i$ that has outgoing degree greater than 1 (otherwise there is only one path from v_i and it is the path to $v_{\max}$) and a path from v_j to a maximal vertex of length $T-j$. Call that path P_j. Let v_j be such a vertex with the largest j. Consider the edge (v_j, v_{j+1}). Since there is no path from v_{j+1} to $v_{\max}$ of length $T-j-1$, there must be a path of length $T-l+j$ from a minimal vertex to v_j. It cannot be the path $v_0, v_1, ...v_j$ since it is shorter than $T-l+j$. Thus, there must be a vertex v_h, $h \leq j$ with an incoming path of length $T-l+h$. Call that path P_h. Figure 5 illustrates the situation. But consider the following path: P_h, followed by the path $v_h, v_{h+1}, ..., v_j$, followed by P_j. The total length of this path is $(T-l+h)+(j-h)+(T-j) = 2T-l$. Since $l < T$, then $2T-l > T$, which is a contradiction to the fact that T is the length of the longest path in G. Thus, every path from a minimal to a maximal vertex in G_T is of length T. Together with the fact that every edge from a path of length T in G is contained in G_T we conclude that G_T contains all and only paths of length T from a minimal to a maximal vertex in G. □

Lemma 3. *Let $G_T + \{s,t\}$ be the graph G_T with the addition of a vertex s connected to every minimal vertex in G_T and a vertex t connected from every maximal vertex in G_T. Then $U \subset V$ is a vertex cut in $G_T + \{s,t\}$ if and only if $G-U$, the subgraph of G induced by $V-U$, has no paths of length T.*

Proof. Suppose, to the contrary, there is a path of length T in $G - U$. Since this is also a path of length T in the original graph G, by Lemma 2 it is also a path of length T from a minimal to a maximal vertex in G_T. Since it is a path in $G - U$, it does not pass through any vertex of U. Since s is connected to every minimal vertex and t is connected from every maximal vertex, it is also a path from s to t in $G_T + \{s, t\} - U$. But then by definition U is not a vertex cut in $G_T + \{s, t\}$. This is a contradiction, therefore there are no paths of length T in $G - U$.

For the other direction, suppose, to the contrary, that U is not a vertex cut in $G_T + \{s, t\}$. Then there is a path from s to t that does not pass through the vertices in U. That path is also a path from a minimal to a maximal vertex in G_T, which by lemma 2 is a path of length T in G. Since it does not pass through the vertices of U, it is also a path of length T in $G - U$, which is also a contradiction. □

We now return to the proof of Theorem 3. From Lemma 3, U is a vertex cut in $G_T + \{s, t\}$ if and only if $G - U$ has no paths of length T. Therefore the algorithm Min T-Path Vertex Shattering Set returns the smallest subset of vertices whose removal from G leaves no paths of length T. Since a minimum vertex cut in a DAG can be computed in polynomial time [12], it is a polynomial time algorithm. □

So far, we have focused on the goal of breaking *all* the metagroups in the population by removing the minimal number of groups. However, when resources are constrained, we may be able to remove only a limited number of groups, up to k. The goal is then to find k groups to remove so that the remaining number of metagroups is minimized. We shall call this problem Critical k-Group Set. Of course, Critical k-Group Set and Critical Groups Set are related.

Theorem 4. Critical k-Group Set *is no easier than* Critical Groups Set.

Proof. Given an algorithm that solves the Critical k-Group Set in time f (for a given α and β) there is an algorithm that solves the Critical Groups Set problem in time $f \log m$ (where m is the total number of groups) for the same α and β. Clearly, we can use a binary search on the limit of groups to be removed as a parameter to the Critical k-Group Set. The smallest number k such that the removal of k-group set leaves no metagroups is the answer to the Critical Groups Set. □

3.7 Critical Individual Set

Another way to address the question of the fragility of a social network is to ask what is the smallest number of *individuals* whose absence would leave no recognizable social structures. In the epidemiological setting, for example, vaccination of those individuals would prevent the spread of a disease (with the right definitions of a group and a metagroup). Critical Individuals Set: Find the smallest set of individuals whose removal leaves no metagroups.

First we note that removing individuals from the population does not guarantee that the metagroups become less persistent (the paths in the metagroups graph become shorter). In

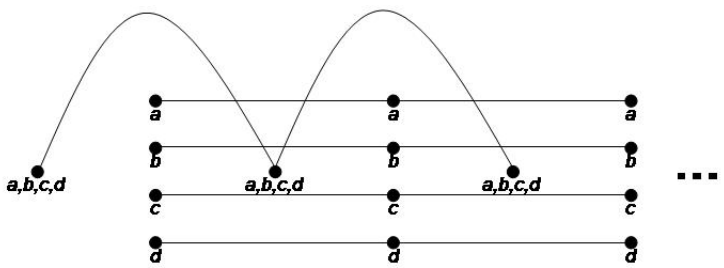

Figure 6. In this graph we consider two groups similar only if they are identical. The removal on any individual from the population does not change the overall structure of the graph until there is only one individual left. Until that point the paths connecting the single individuals do not intersect with the path connecting the group of all the individuals and all are $T/2$ long. When there is only one individual left then there is only one path remaining and it is T long.

fact, Figure 6 shows an example of a metagroups graph for which the removal of all but one individual (no matter which one) leads to a doubling of the path length. While the example in Figure 6 may seem contrived, in fact the same phenomenon occurs in *every* metagroups β-graph where β is such that only identical groups are connected. Since only identical groups are connected then the removal of any individual does not make identical groups different and thus disconnect them. In fact, the removal of the individuals that make up the difference between two groups will cause those groups to be connected where they were not before. Any such "identical only" metagroups graph initially consists of cliques of identical groups. As the individuals are removed, some cliques disappear as all their members are eliminated, while others are joined, as the differences between them are removed, until there is only one individual left and the corresponding metagroup path is T long. If all the cliques in the initial such graph are of size α or greater (recall that the goal is to remove individuals until there are no metagroups, i.e. no paths of length α or greater) then there is no set of individuals that can be removed to leave the graph with paths less than α. However, all of this structure hinges on the fact that the only similar groups are the identical ones. *Thus, it is very hard to break the social structure of the identical groups.*

The example above shows that the identity similarity measure may be impractical, particularly in the context of disease or information spreading. The identity similarity measure assumes that even groups that share most of their members but are not identical do not pass information between them or transmit diseases. Once we relax the requirement for the two groups to be identical, both the realistic implications and the combinatorics of the critical

individual set change drastically.

Proposition 3. *For any edge in the graph (h, g) and any individual x there are three possible ways in which the removal of x can affect the weight of the edge (h, g):*

1. *If $x \notin h$ and $x \notin g$ then $sim(h, g)$ does not change with the removal of x.*
2. *If $x \in h \cap g$ then $sim(h, g)$ will not increase with the removal of x, but may decrease.*
3. *If $x \in h$ but $x \notin g$ (or vice versa) then $sim(h, g)$ will not decrease with the removal of x, but may increase.*

Proof. Recall that we assume that the similarity measure has the following three properties:

1. $sim(h, g)$ depends only on h and g.
2. $sim(g, g) =$ maximum similarity value.
3. $sim(h, g)$ monotonically increases with the increase of $|h \cap g|$ when $|h| + |g|$ does not change.
4. $sim(h, g)$ monotonically increases with the decrease of $|h| + |g|$ when $|h \cap g$ does not change.

Since $sim(h, g)$ depends only on the values of h and g, then if an individual x is in neither group its removal does not affect the values of those groups and, hence, $sim(h, g)$ does not change.

If an individual $x \in h \cap g$ then with the removal of x the intersection $h \cap g$ decreases by 1 while $|h| + |g|$ decreases by 2. By the similarity measure properties 3 and 4 $sim(h, g)$ may decrease with the removal of x, but will not increase.

If an individual x is in one of the groups h or g but not in both then with the removal of x the intersection $h \cap g$ does not change while $|h| + |g|$ decreases. Hence, by the similarity measure property 4 $sim(h, g)$ may increase with the removal of x, but will not decrease. □

Thus, a possible greedy strategy is to remove the individuals that appear in the intersections of the largest number of groups that are still connected by an edge. We give the precise algorithm below.

Alg GREEDY CRITICAL INDIVIDUALS SET

$S = \emptyset$
WHILE there are paths of length $\geq \alpha$ **DO**
    Let $x \in X$ be such that $|\{(h,g) \in E, \text{ s.t. } x \in h \cap g\}| = \max_{y \in X} |\{(h,g) \in E, \text{ s.t. } y \in h \cap g\}|$
    FOR all $e = (h,g) \in E$ s.t. $x \in h \cap g$ **DO**
        $h = h - \{x\}, g = g - \{x\}$
        compute $w(e) = sim(h,g)$
        IF $w(e) < \beta$ **THEN**
            $E = E - \{e\}$
    $X = X - \{x\}$
    $S = S \cup \{x\}$
RETURN S

There many other possible greedy strategies one may consider:

1. remove x that removes the largest number of edges
2. remove x that reduces the (total) weight of the edges by most
3. remove x that removes the heaviest edges
4. remove x that removes edges from the largest number of metagroups

We merely list these strategies here and leave the detailed investigation for the future.

Again, we may consider the dual problem of minimizing the remaining number (or length, or size) of the metagroups while removing at most k individuals, CRITICAL INDIVIDUALS k-SET. As in the standard parametrized approach, the problem is no easier than the general one and adds at most a logarithmic factor to any general algorithm.

We now present a pseudo-polynomial algorithm for a fixed k. For any given set of individuals it is straight forward to calculate the new weight for every edge in the graph if that set is removed from the population. Therefore, for any fixed set of individuals it takes at most $O(l^2 f(\min\{n-k,k\}))$ time to compute the new meagroups β-graph (where l is the number of groups in the graph, n is the number of individuals in the original population and k is the size of the proposed set to be removed.) Here we assume that $f(\min\{n-k,k\})$ is the time it takes to adjust the similarity measure for an edge.

4 Extensions and Open Problems

There are many other questions one may ask about a dynamic society that can be easily modeled computationally in our framework. We give a list of open problems here.

- CLOSEST FRIENDS: Find a set of k individuals that appear together (not necessarily the same metagroup) more than any other set of k individuals.

- Individual Membership: Given an individual x, find the metagroup MG which maximizes the cardinality of the set of groups in MG in which x occurs. That is, $|\{i,\ s.t.\ x \in g_i,\ g_i \in MG\}| = \max_{\text{metagroups } D} |\{j,\ s.t.\ x \in g_j,\ g_j \in D\}|$. We note that in some cases, we may be concerned with maximizing frequency rather than the absolute number. Alternatively, find the most persistent or the largest metagroup MG of which x is a member (same as Individual Connectivity with $S = \{x\}$).

- Extraverts and Intraverts: Find the individual who is a member of the largest (smallest) number of metagroups.

- Loyal Individuals: Given an individual, what fraction of its time is it a member of the same metagroup? Find the individuals that appear most frequently in any metagroup (among all the metagroups).

- Metagroup Representative: Given a metagroup MG, is there an individual who occurs more in this metagroup than any other individual and occurs in MG more than in any other metagroup?

- Demographic Distinction: Given a coloring of individuals (a partition), is there a property that distinguishes one color from the others, i.e. some color is in more metagroups, fewer metagroups, longer metagroups, more time in any metagroup (e.g. on average), etc.? Each color represents a demographic set, such as lactating females, juveniles, computer scientists, cellphone users, etc.

- Critical Parameter Values: Identify the largest values of α, β for which there exists at least 1 (k) metagroup. Identify the largest value of γ for which each metagroup has at least one member.

- Sampling Rate: Identify the appropriate sampling rate with respect to the posted question, i.e. the largest time step such that the answer does not change if the time step is subdivided but does change if it is increased.

- Critical Time Moments: Identify the critical time moments. For example, the time when the groups' membership changes most, i.e. minimal edge weight sum between time steps. Another version is to find the most connected (important?) individuals/groups as a function of time. In this case there may be one time step when everybody comes together and that is a critical time point for vaccination, for example.

- Data Augmented Solution Reconciliation: Given partial sets of observations (with some groups missing), such that their union comprises the entire set of groups, and given partial solutions to any of the above problems based on each set, what is the combined solution to the entire input?

For all of the above questions it is important also to keep in mind that the input graph may be very large and even polynomial algorithms may be too slow.

Acknowledgments

We are grateful to S. (Muthu) Muthukrishnan, Dan Rubenstein, Siva Sundaresan, Ilya Fischoff, David Kempe, Simon Levin and many others for their help in designing this dynamic social analysis framework.

This work is supported by the National Science Foundation postdoctoral fellowship grant EIA 02-05116 (Tanya Berger-Wolf), by the National Science Foundation grant CCR-0313160 (Jared Saia), and by the Sandia University Research Program Grant No. 191445 (Jared Saia).

References

[1] ***. ***. ***, ***.

[2] E. Airoldi and B. Malin. Data mining challenges for electronic safety: the case of fraudulent intent detection in e-mails. In *Proceedings of the Privacy and Security Aspects of Data Mining Workshop, in conjunction with the 4th IEEE Internation Conference on Data Mining*, Brighton, England, 2004.

[3] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. Discovering hidden groups in communication networks. In *Proceedings of the 2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, 2004.

[4] S. Bernstein, A. Clearwater, S. Hill, C. Perlich, , and F. Provost. Discovering knowledge from relational data extracted from business news. In *Proceedings of the Workshop on Multi-Relational Data Mining, in conjunction with the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[5] A. Broido and K. Claffy. Internet topology: connectivity of ip graphs. In *Proceedings of SPIE ITCom*, 2001.

[6] K. Carley. Communicating new ideas: The potential impact of information and telecommunication technology. *Technology in Society*, 18(2):219–230, 1996.

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

[8] L. Chen and K. Carley. The impact of social networks in the propagation of computer viruses and countermeasures. *IEEE Trasactions on Systems, Man and Cybernetics*, forthcoming.

[9] D.P Croft, J. Krause, and R. James. Social networks in the guppy (poecilia reticulata). *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 271:516–519, 2004.

[10] Paul C. Cross, James O. Lloyd-Smith, and Wayne M. Getz. Disentangling association patterns in fissionfusion societies using african buffalo as an example. *Animal Behaviour*, 69:499–506, 2005.

[11] S. Eubank, H. Guclu, V.S. Kumar, M.V. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429:429:180–184., Nov 2004. Supplement material.

[12] S. Even and R.E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4:507–518, 1975.

[13] N. Hummon and K. Carley. Social networks: As normal science. *Social Networks*, 15:71–106, 1993.

[14] P. Jaccard. The distribution of flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912.

[15] M. Keeling. The effects of local spatial structure on epidemiological invasions. *Proc. R. Soc. Lond. B*, 266:859–867, 1999.

[16] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[17] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the Thirty-second ACM Symposium on Theory of Computing (STOC)*, 2000.

[18] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence,*, Morgan Kaufman, 2001.

[19] J. Kleinberg. The small-world phenomenon and decentralized search. *SIAM News*, 37, 2004.

[20] Gina Kolata. Ideas and trends; enron offers an unlikely boost to e-mail surveillance. *New York Times*, May 22 2005.

[21] M. Kretzschmar and M. Morris. Measures of concurrency in networks and the spread of infectious disease. *Math. Biosci.*, 133:165–195, 1996.

[22] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proc. 20th Int. Colloquium on Automata*, volume 700 of *Lecture Notes in Comput. Sci.*, pages 40–51. Springer-Verlag, 1993.

[23] D. Lusseau and M. E. J. Newman. Identifying the role that individual animals play in their social network. *Proc. R. Soc. London B (Suppl.)*, 271:S477–S481, 2004.

[24] Newman M. and Girvan M. Finding and evaluating community structure in networks. *Phys. Rev.*, 69, 2004.

[25] M. Magdon-Ismail, M. Goldberg, W. Wallace, and D. Siebecker. Locating hidden groups in communication networks using hidden markov models. In *Proceedings of the International Conference on Intelligence and Security Informatics (ISI 2003)*, Tucson, AZ., 2003.

[26] B. Malin. Data and collocation surveillance through location access patterns. In *Proceedings of the North American Association for Computational Social and Organizational Science (NAACSOS) Conference*, Pittsburgh, PA, June 2004.

[27] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[28] M. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev.*, 69, 2004.

[29] C. Papadimitriou. Computational aspects of organization theory. *Lecture Notes in Computer Science*, 1997.

[30] C. Papadimitriou and E. Servan-Schreiber. he origins of the deadline: Optimizing communication in organizations. *Complexity in Economics.*, 1999.

[31] M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying. Customer coalitions in electronic marketplaces. In Ulises Cortes Frank Dignum, editor, *Agent-Mediated Electronic Commerce III, Lecture Notes on Artificial Intelligence*. Springer-Verlag, 2003.

[32] J.R. Tyler, D.M. Wilkinson, and B.A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Proceedings of the First International Conference on Communities and Technologies*, 2003.

[33] S. Wasserman and Faust K. *Social Network Analysis.* Cambridge University Press, Cambridge, MA, 1994.

[34] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[35] B. Wellman. An electronic group is virtually a social network. In Sara Kiesler, editor, *Culture of the Internet*, pages 179–205. Lawrence Erlbaum, Mahwah, NJ, 1997.