Practical Issues and Algorithms for Analyzing Terrorist Networks[1]

Tami Carpenter, George Karakostas, and David Shallcross
Telcordia Technologies
445 South Street
Morristown, NJ 07960

Keywords: social network analysis, centrality, betweenness, path-finding algorithms.

INTRODUCTION

In social network analysis, graphs are used to model relationships between actors or participants in a social setting. Each node or vertex in the graph represents a participant or actor. Each link or edge represents a connection or relationship between two participants. A variety of graph algorithms have been developed to analyze the structure of social networks and to assess the roles or importance of the individual players. Since the September 11 bombing of the World Trade Center, social network analysis has emerged as a potential vehicle for modeling and analyzing the structure of terrorist networks [10, 15].

There are a variety of measures to assess the "importance" or *centrality* of each actor in a social network [16]. The most popular of these centrality measures require the computation or enumeration of shortest paths between all pairs of nodes in the graph. Such computations can be time consuming in large graphs. Moreover, they may become problematic even in more moderately-sized networks when changing data or "what-if" scenario analysis warrant frequent recomputation.

In this paper, we assume that terrorist network models may be large, dynamic and characterized by uncertainty. The latter two properties, in particular, are true not only for terrorist networks, but are true more generally of "covert" networks, where there may be a deliberate effort to hide illicit activity. Models of covert networks may be large, not necessarily because the networks themselves are large, but because the networks are unknown to us, so the set of actors we monitor is likely to be a superset of those that are actually engaged in illicit activity. The models are dynamic because both covert networks and our knowledge of them change over time. Likewise, specific attributes associated with nodes or links in the network model contain uncertainty. Although social network analysis has been used to study covert criminal networks in the past (see [5] for example), these networks are not as large or as dynamic as terrorist networks, nor are the stakes surrounding them as high.

A previous paper in this session (see Behrens and Stephenson) describes a framework (called *NetOperative*) to support lawmakers in their effort to identify and monitor terrorist networks. This framework combines statistical techniques with ideas from social network analysis and with efficient computational algorithms to support such analysis. In this paper we focus on algorithms for computing measures of centrality. We describe the current best theoretical running times for computing various centrality measures; we discuss the current state-of-the-art for dynamically updating these measures in response to network changes; and we describe some practical decomposition ideas that can also benefit from maintaining dynamically updated data structures. In addition, we discuss some alternative measures of centrality that may be more robust in the face of uncertainty.

CENTRALITY MEASURES

To define centrality measures more precisely, we need to establish some notation. The notation that we employ is similar to that defined in Brandes [3]. We let $G=(V, E)$ denote a graph with vertex set V and edge set E. In general, edges may be either directed or undirected. We let n represent the number of vertices in V, and we let m represent the number of edges in E. An edge $e \in E$ has an associated weight or length denoted $l(e)$. If the graph is "unweighted", $l(e)$ is assumed to be equal to 1 for all $e \in E$. Edges may be represented as a pair of incident vertices, so that (s,t) represents the edge between nodes $s, t \in V$. We let $d(s, t)$ denote the shortest

[1] Copyright 2002, Telcordia Technologies, Inc. All rights reserved.

possible distance between vertices *s* and *t* in *G*. Finally, we let σ_{st} denote the *number* of shortest paths between *s* and *t* in *G*, and we let $\sigma_{st}(v)$ denote the number of such paths that include vertex $v \in V$.

Now, we can define two well-known [16] centrality measures that we use for illustrative purposes in subsequent discussion. Additional measures and variations are described in the text by Wasserman and Faust [16] and references therein.

$$C(v) = \frac{1}{\sum_{t \in V} d(v,t)} \qquad \textit{Closeness centrality,}$$

$$B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad \textit{Betweenness centrality.}$$

We note that computing closeness centrality requires computing the *length* of a shortest path between all pairs of nodes in *G*. This requires solving the well-known *all-pairs shortest-paths* (APSP) problem. The current best algorithms for APSP on sparse graphs require $O(nm)$ operations in unweighted graphs and $O(nm + n^2 \log n)$ operations in weighted graphs [9]. In dense graphs, these bounds asymptotically become $O(n^3)$ in the worst case, but recent (more complicated) algorithms yield somewhat better worst-case bounds [18]. Computing betweenness requires enumerating *all* of the shortest paths between each pair of nodes, which is even more work. The fastest algorithms for computing betweenness are provided by Brandes [3] and require $O(nm)$ operations in unweighted graphs and $O(nm + n^2 \log n)$ operations in weighted graphs. These bounds are asymptotically the same as those for solving APSP in sparse graphs. Empirical results presented in [3] for randomly-generated, undirected, unweighted, sparse graphs suggest that betweenness in a 6000 node graph can be computed in roughly 15 minutes on a SUN Ultra10 workstation. Solution times appear to be slower for weighted graphs and will slow considerably as *n* increases into the tens of thousands of nodes.

PRACTICAL ATTACKS ON COMPUTING CENTRALITY

As *n* gets very large, the time to compute the centrality measures can become prohibitive. Nonetheless, there are some practical methods that can sometimes be exploited to make these computations more tenable in large graphs. Thus, in addition to seeking algorithms with improved theoretical running times for the underlying problems like APSP, we can try to exploit other features characteristic of our application. In particular, the sparsity of social networks may be exploited in practical approaches that involve approximation or decomposition. Often these strategies are employed in attacks on NP-hard optimization problems, but they can be applied any time the size of a problem makes it difficult to solve.

Given the amount of uncertainty that we expect to encounter in models of terrorist networks, approximation methods may be particularly attractive for these applications. A recent paper by Eppstein and Wang [8] exploits the "small world" phenomenon that is said to characterize social networks in order to develop an approximation algorithm for closeness centrality. An open question is: *can we develop fast approximation algorithms for betweenness or for the more closely related graph centrality measure* [3], which is defined as:

$$g(v) = \frac{1}{\max_{t \in V} d(v,t)} \qquad \textit{Graph centrality.}$$

More common practical approaches involve decomposition. Sparse networks may have "small cuts" that yield places where we can break the problem apart. (A cut in a graph is a set of vertices or edges whose removal disconnects the graph.) A basic decomposition strategy would proceed as follows: identify a cut; break the problem apart at the cut; solve the smaller problems on each side of the cut; and "sew" the solutions together to obtain a solution on the entire graph. This strategy is often employed in attacks on NP-hard optimization problems. (See, for example, [4].) However, even for computationally tractable problems, like APSP, we might still use decomposition in a heuristic fashion to effectively reduce *n* in the running time.

The simplest example of decomposition at a small cut arises in undirected graphs where we consider breaking the problem apart at articulation points. *Articulation points* are single vertices whose removal disconnects the graph. We say that two vertices are *biconnected* or lie in the same *biconnected component* if there is no articulation point whose removal disconnects them. If a node *v* is an articulation point, removing *v* breaks the graph into (at least) two components, which we call *S* and *T* in Figure 1. Any shortest path between nodes in $S \cup \{v\}$ must involve only nodes in $S \cup \{v\}$. Likewise, shortest paths between nodes in $T \cup \{v\}$ involve only nodes in $T \cup \{v\}$. Thus, in Figure 1, shortest paths between nodes in $S \cup \{v\}$ use only gray links and shortest paths between nodes in $T \cup \{v\}$ use only black links. Paths between two nodes, $s \in S$ and $t \in T$, are constructed as the concatenation of shortest paths between *s* and *v* with shortest paths between *v* and *t*. Thus, we can find all shortest paths by solving a smaller problem on each "side" of *v*, effectively reducing *n* in the running time.

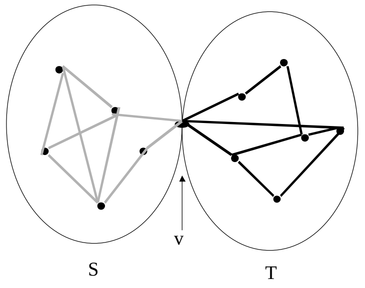

Figure 1: Decomposing at articulation point v.

An attractive aspect of decomposing at articulation points is that articulation points and the shores induced by their removal are identified by a simple $O(m)$ algorithm [1]. The decomposition idea can be generalized to other types of small cuts, like two-edge cuts, but more general cuts are more time-consuming to identify and also more difficult to reconstruct solutions across. Even more general decomposition strategies may also be available. The *tree width* of a graph may be thought of as a measure of its complexity, and graphs with small tree width can be recursively decomposed by removing small sets of nodes. Such an approach requires sophisticated ideas from graph theory surveyed in [2]. Empirical studies are needed to see whether these more complicated ideas yield practical savings.

CENTRALITY IN DYNAMIC GRAPHS

In addition to being large and sparse, terrorist networks, or at least our models of them, are likely to be dynamic through time, with new links being added and others being removed as we gain information about the actors and their communications. Thus, centrality measures may have to be recomputed as the network evolves through time. In addition to reducing the time spent on the initial calculation, we would also like to identify "cheap" algorithms for updating centrality measures following a change in the network. Such algorithms can also be of use in "scenario analysis", wherein we may want to examine the effect of potential network changes. For instance, within a tool like NetOperative, an analyst could examine the sensitivity of the centrality measures to "likely" network changes, such as the addition of a believed link or the removal of a key node.

Finding efficient algorithms to update the solution of a graph problem following the addition or deletion of links is an area of intense research in the computer science community, as described in a recent survey by Eppstein, Galil, and Italiano [7]. Since the centrality measures we consider involve the calculation of shortest paths between all pairs of nodes, *dynamic all pairs shortest path algorithms* may help us avoid recalculating these paths from scratch. For example, the algorithm by Demetrescu and Italiano [6] supports updates (edge deletion, insertion, and weight change) in $O(Sn^{2.5} \log^3 n)$ time per update in directed networks with weights that take at most *S* different real values.

Dynamic algorithms for computing other graph properties that may be helpful in carrying out the centrality calculations more efficiently are also described in the survey [7]. For example, the minimum spanning tree for undirected graphs can be maintained in time $O(n^{1/3} \log n)$ per update. 2-edge connectivity can be solved in

$O(n^{1/2})$ time per update; 3-edge connectivity can be solved in $O(n^{2/3})$ time per update. For k-vertex connectivity with $2 \le k \le 4$ there are algorithms with times ranging from $O(n^{1/2} \log^2 n)$ to $O(n\alpha(n))$ per update. More details can be found in [7].

If we use a decomposition-based approach, we may, in fact, wish to maintain data structures that allow us to update our knowledge of cuts and shores of cuts without having to fully recompute them. One of the best-studied of these problems is for fully dynamic biconnectivity [11] in undirected graphs. A fully dynamic biconnectivity algorithm maintains and updates a data structure that allows efficient queries about the biconnected components of a dynamic graph. The algorithm described by Henzinger [11] performs updates in $O(\sqrt{m}\log n)$ time and answers queries about whether or not two nodes are in the same biconnected component in constant time. Further, it returns all of the nodes in a biconnected component in time that is linear in the size of the output. Such an algorithm could be integrated within a larger framework for monitoring dynamic social networks. Finally, as mentioned above algorithms to support other connectivity queries in dynamic graphs have also been the subject of recent study [7]. (See, also [13], which considers 2- and 3-edge connectivity with link insertions.)

EXTENDING BETWEENNESS

The measures of centrality that we defined previously (along with several other well-known measures [16]) implicitly assume that communication occurs along shortest paths (also called geodesic paths) in the network. In analyzing networks that model terrorist activity, uncertainty in the data will be reflected in inaccuracies in shortest path computations. Thus, not only is the length of the shortest path between a pair of nodes somewhat uncertain, but the path itself may change dramatically with relatively small changes in the data. This may be especially problematic for measures like betweenness, that depend on knowing the precise identity of nodes in geodesic paths. The small example in Figure 2 illustrates how the betweenness centrality measure can be vulnerable to even minor data changes.

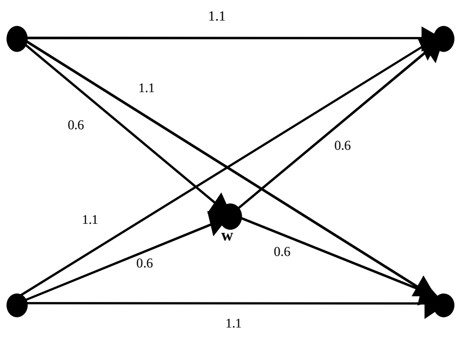

Figure 2: Sample network where B(w) = 0.

With the given edge lengths, the definition of betweenness yields $B(w)$=0, even though node w appears to be relatively "central". Notice that if the actual length of all "big" edges is 1.3 instead of 1.1, w becomes *the* central actor in this network, and $B(w)$=4. Examples like this motivate us to look for more robust variations of the betweenness measure. For this reason, we propose a variation on the betweenness centrality measure, called **ε-betweenness.**

Before we give the formal definition of this new measure and an algorithm to compute it, we give a few more preliminary definitions. First, a *simple* path between nodes s and t is a path connecting s and t that visits no node more than once. A simple path P from s to t will be called an *ε-shortest path* if length(P)≤(1+ε)$d(s,t)$. We let $\sigma_{st}^{\varepsilon}$ denote the *number* of ε–shortest paths between s and t in G, and we let $\sigma_{st}^{\varepsilon}(v)$ denote the number of such paths that include vertex $v \in V$. We define the ε-betweenness ***E(v,ε)*** as follows:

$$E(v,\varepsilon) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}^{\varepsilon}(v)}{\sigma_{st}^{\varepsilon}} \quad \textit{ε-Betweenness centrality.}$$

Notice that in the example given above, with ε=0.1 (i.e. we count paths whose length doesn't exceed that of the shortest path by more than 10%), *E(w,0.1)*=2.

Since the calculation of ε-betweenness involves approximate shortest paths, it is more complicated and time-consuming than the calculation of betweenness. Furthermore, when we consider paths that may be longer than the shortest path, we must take care to exclude paths containing loops. Paths containing loops seem to be unlikely communication paths and are thus explicitly excluded. In this algorithm, we assure that the paths computed are simple by enforcing that ε d(s,t)<2l(e) for all edges e and all pairs $s \neq t$. In particular, if l is the length of the shortest edge and D is the maximum shortest path distance between a pair of nodes in G, then $\varepsilon < 2\left(\frac{l}{D}\right)$.

Now, we give a straightforward algorithm to compute ε-betweenness. Let $\Gamma(t) = \{u \in V : (u,t) \in E\}$ be the nodes that are directly connected to t, and let $w \in \Gamma(t)$. We calculate the number of ε-shortest paths from s to t that go through the edge (w,t). Notice that any path P from s to w with length(P)≤(1+δ)d(s,w), where $\delta = \frac{(1+\varepsilon)d(s,t) - (d(s,w) + l(w,t))}{d(s,w)}$, can be extended with edge (w,t) to a path from s to t, whose length is at most (1+ε)d(s,t). Moreover, $\delta \geq 0$ implies that only the w's that satisfy $(1+\varepsilon)d(s,t) \geq l(w,t) + d(s,w)$ can produce such paths. It is obvious that, if we calculate δ_i for all neighbors w_i of t that satisfy the last condition, $\sigma_{st}^{\varepsilon}$ can be computed as follows:

$$\sigma_{st}^{\varepsilon} = \sum_i \sigma_{sw_i}^{\delta_i} .$$

Assuming that we have already calculated the shortest path lengths (by, say, the method in [9]), $\sigma_{st}^{\varepsilon}$ can be computed recursively, searching from t in a breadth-first fashion. In order to calculate $\sigma_{st}^{\varepsilon}(v)$, we store all intermediate results $\sigma_{sw_i}^{\delta_i}$. Then it is easy to retrieve the number of paths that pass through v.

The running time of this algorithm can be prohibitively high when the network is not acyclic. However, it is conceivable that the condition $(1+\varepsilon)d(s,t) \geq l(w,t) + d(s,w)$ together with the rapid convergence of δ's to 0 may keep the space and time requirements of the algorithm manageable. This is certainly a point for further investigation.

An algorithm for ε-betweenness can also be implemented based on a variant of a k-shortest paths algorithm, for computing the k shortest acyclic paths between a pair of nodes. Given a graph with weights on the edges, a pair of vertices s and t, and an integer k, such an algorithm computes the shortest s-t path, the second shortest s-t path, up to the kth shortest s-t path. Several algorithms, including those by Katoh, Ibaraki, and Mine [12] and Yen [17], compute these paths in order, so that we don't have to decide on k in advance, but rather can proceed until we generate a path of length greater than (1+ε)d(s,t). At this point we have generated all of the ε-shortest s-t paths, and we can easily determine both $\sigma_{st}^{\varepsilon}$ and $\sigma_{st}^{\varepsilon}(v)$ for all vertices v. The time to do this in undirected graphs using the algorithm in [12] is $O((\sigma_{st}^{\varepsilon}+1)c(n,m))$, where $c(n,m)$ is the time to compute a shortest-path tree from a single vertex in a graph with n vertices and m edges. This algorithm works for arbitrary nonnegative ε.

Underlying the definition of ε-betweenness is the assumption that our betweenness measure will be more stable in the presence of uncertain data if we base it on a larger set of "good" paths. As such, we could also construct a betweenness variant based upon the k-shortest simple paths. However, we prefer the ε-betweenness measure because it is more sensitive to the *relative* quality of the paths for each pair of nodes.

These more robust measures are predicated on the assumption that communication still occurs along geodesic paths in the network, but we don't necessarily *know* those paths because of uncertainty in the data. Stephenson and Zelen [14] introduced another measure of centrality, called *information centrality*, that weighs all paths between a pair of nodes. Their thinking is that *all* paths carry information. Moreover, circuitous paths may be preferred when actors wish to hide their communications. This measure also has the robustness of considering multiple paths and may be very well-suited to analyzing terrorist networks where deliberate efforts are made to obfuscate communication.

FUTURE DIRECTIONS

We have described both theoretical and practical avenues of research for computing centrality in covert networks. Perhaps the most compelling theoretical questions are whether running times for dynamic APSP can be improved for sparse networks and whether such algorithms can be extended to computing betweenness. Given the amount of uncertainty that we expect in models of terrorist networks, finding fast approximation algorithms for betweenness and other centrality measures is also of interest. From a practical standpoint, it is of interest to see which types of small cuts we find in real networks, whether social networks have small tree width, whether decomposition to exploit these features speeds centrality computation, and whether more robust measures of centrality are needed.

Reference List

[1] Aho, J. Hopcroft, and J. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[2] D. Bienstock and M. Langston. Algorithmic implications of the graph minor theorem. In *Network Models*, Chapter 8, M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, eds., Elsevier Science, Amsterdam, 1995.

[3] U. Brandes. A faster algorithm for betweenness centrality. To appear in *Journal of Mathematical Sociology*.

[4] G. Cornuejols, D. Naddef, and W. Pulleyblank. The travelling salesman problem in graphs with 3-edge cutsets, *J. ACM*, 32, pp. 383-410, 1985.

[5] R. Davis. Social network analysis: An aid in conspiracy investigations. *FBI Law Enforcement Bulletin*, 50(12), pp. 11-19, 1981

[6] C. Demetrescu and G. Italiano. Fully dynamic all pairs shortest paths with real edge weights. *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pp. 260-267, Las Vegas, 2001.

[7] D. Eppstein, Z. Galil, and G. Italiano. Dynamic graph algorithms. In *Algorithms and Theoretical Computing Handbook*, Chapter 8, M. J. Atallah, ed., CRC Press, 1999.

[8] D. Eppstein and J. Wang. Fast approximation of centrality. *Proceedings of the 12th ACM Symposium on Discrete Algorithms*, pp. 228-229, Washington, 2001.

[9] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34, 596-615.

[10] J. Garreau. Disconnect the dots: maybe we can't cut off terror's head but we can take out its nodes. *Washington Post Online*, September 16, 2001.

[11] M. Henzinger. Improved data structures for fully dynamic biconnectivity. *SIAM Journal on Computing*, 29, pp. 1761-1815, 2000.

[12] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for K shortest simple paths. *Networks* 12(4), pp. 411-427, 1982.

[13] H. La Poutre. Maintenance of 2- and 3-edge-connected components of graphs II. *SIAM Journal on Computing*, 29, pp. 1521-1549, 2000.

[14] K. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11, pp. 1-37, 1989.

[15] T. Stewart. Six degrees of Mohammad Atta. *Business 2.0*, December, 2001.

[16] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[17] J. Y. Yen. Finding the K shortest loopless paths in a network. *Management Science* 17, pp. 712-76, 1971.

[18] U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. Online technical report.

Biography

Tami Carpenter received her Ph.D. in Operations Research from Princeton University in 1992. Upon completion, she joined Bellcore (now Telcordia Technologies) and is director of the Network Optimization and Algorithms Research Group. She conducts research in communication network optimization.

George Karakostas received his Diploma in Computer Engineering and Informatics from the University of Patras in 1995. He got his Ph.D. in Computer Science from Princeton University in 2000. In 2000, he joined Telcordia Technologies. His research interests include the design and analysis of approximation algorithms, computational complexity, and practical applications of theory.

David Shallcross received his Ph.D. in Operations Research from Cornell University in 1989. After postdoctoral positions at Yale University and at IBM, he joined Bellcore (now Telcordia Technologies) in 1992. His work has been in combinatorial and network optimization.