The use of sparsest cuts to reveal the hierarchical community structure of social networks

Charles F. Mann a,*, David W. Matula a, Eli V. Olinick b

a Department of Computer Science and Engineering, Southern Methodist University, United States
b Department of Engineering Management, Information and Systems, Southern Methodist University, United States

A R T I C L E   I N F O

Network structure
Community structure
Social networks
Maximum concurrent flow
Sparsest cut
Multipartite cut
Divisive algorithm
Graph theory

A B S T R A C T


The MCFP extends the [Ford Jr., Lester R., Fulkerson, Delbert R., 1956. Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404; Ford Jr., Lester R., Fulkerson, Delbert R., 1962. Flows in Networks. Princeton University Press, Princeton, NJ] source–sink max-flow problem to all-pairs maximum concurrent flow. The density of an (S, T)-cut is |(S, T)|/|S|•|T| where |S, T| is the number of edges (equivalently, links or bonds) between communities S and T with |S|•|T| being the maximum number of edges possible. The minimum density cut in the network is the sparsest cut. When the edges are weighted, the density is the average weight of the (S, T)-cut edges, with absent edges treated as edges of zero weight. Sparsest cuts (i.e., minimum density) of the remaining components are iteratively determined via linear programming until a multipartite cut is identified that is more constraining to concurrent flow than any sparsest cut.

Empirical results on real-world networks with known hierarchical structures and random networks with embedded communities are used to compare this sparsest-cut community structure criterion to the weighted Girvan–Newman community structure criterion [e.g., Newman, M.E.J., 2004. Analysis of weighted networks. Physical Review E 70, 056131] that is based on edge betweenness centrality. A new measure of accuracy M is defined to evaluate the results of the graph partitionings found by these criteria.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Blockmodeling is a well-established technique used by many researchers in the field of social network analysis to find the community structure via a sophisticated permutation of the rows and columns of the network’s adjacency matrix (Doreian et al., 2005). It is a matrix-based methodology developed in the 1970s by Lorrain, White, Breiger, Boorman, and others (Lorrain and White, 1971; Breiger et al., 1975; White et al., 1976) for grouping the members of social networks into blocks of individuals who share a common
pattern of ties both within and between other blocks of the network. Doreian et al. (2005) developed a generalized approach to blockmodeling going beyond structural characteristics to use other relations between actors to fit a generalized blockmodel. Blockmodeling performs a positional analysis identifying the role and position of individuals.

Hierarchical approaches to this problem are connectivity analyses of the network that identify the structural elements. The hierarchical methods fall into either of two major approaches known as agglomerative and divisive (e.g., Cole, 1969; Cox et al., 1993; Matula, 1986; Sneath and Sokal, 1973). The agglomerative approach is a bottom-up approach that has been traditionally favored due to its computational tractability. Agglomerative methods iteratively connect actors (called vertices) in a social network that are close to each other property-wise with ties or links (called edges). Small closely bound subcommunities are merged with each pass building up the hierarchical structure to find the desired larger community structure. However, atypical local associations among the starting elements can lead to larger structures that are not the most homogeneous communities. This can result in erroneous conclusions about the community structure of the network.

To avoid the deficiencies associated with agglomerative clustering methods, graph theoretic divisive methods based on network flows (e.g., Jardine and Sibson, 1971; Matula, 1977, 1983, 1985, 1986) and based on edge centralities (e.g., Girvan and Newman, 2002; Newman, 2004; Fortunato et al., 2004) have been developed. The edge centrality methods iteratively identify the inter-community edges in a top-down approach which, when removed, reveal the community structure of the network. The Girvan–Newman method uses edge betweenness centrality and the Fortunato–Latora–Marchiori method uses edge information centrality as the criterion for edge removal.

The divisive algorithm investigated in this paper is based on cut density and maximum concurrent flow. We will therefore refer to it as the MCF cut algorithm. It iteratively solves the maximum concurrent flow problem (MCFP) determining sparsest cuts until a multipartite cut is identified that is more constraining to concurrent flow than any sparsest cut. Sparsest cut identifies sets of inter-community edges that partition the network to reveal a hierarchy of community structures. Sparsest cuts are a measure of the density, or capacity, of the partitioning edge sets. Sparsest cuts are visualized as sparse, off-diagonal rectangular regions in a permuted adjacency matrix where communities are arranged in sparse blocks straddling the main top-left to bottom-right diagonal. Historical work by researchers employing density measures to find community structure in social networks has been noted by Scott (2000).

In contrast to the density-based methods for finding community structure are those that are based upon measures of centrality. Borgatti and Everett (2006) observe that density is a whole-network measure of cohesion that is like centrality measures except that centrality summarizes the cohesion by vertex. Based on flow on shortest paths, but not making direct use of density, the Girvan–Newman (GN) algorithm (e.g., Girvan and Newman, 2002; Newman, 2004) employs all-pairs shortest path counts to assign edge betweenness centrality values to each edge. The GN algorithm iteratively removes the one or more edges with the highest edge betweenness centrality value at each iteration and then recomputes the shortest path counts and edge betweenness centrality values removing further highly used edges until a partition is found in the remaining network. This sometimes partitions off low-level communities without recognizing the high-level hierarchical groupings, and sometimes prematurely removes edges that are not a part of the next hierarchical partition.

For randomly generated, weighted networks, the weighted version of the Girvan–Newman algorithm tends to be more accurate than the MCF cut algorithm on sparse graphs where the number of inter-community edges is less than the number of intra-community edges. However, in dense graphs, as the ratio of inter- to intra-community edges increases, the MCF cut algorithm becomes more accurate than the GN algorithm. This is particularly true for graphs where most edges are weighted. The minimum density cut is then the partition with the minimum average weight on all edges of the cut with any absent edges treated as having zero weight.

The remainder of this paper is structured as follows. In Section 2 we formally define sparsest cuts and demonstrate how they determine a hierarchical decomposition of a graph. The foundations for this hierarchical decomposition are presented and provide the motivation for our proposed algorithm for determining community structures in social networks. In Section 3 we discuss the MCFP and describe how our decomposition algorithm uses it to find sparsest cuts. In Section 4 we present a general measure for the similarity of two network partitionings. We present empirical results of the algorithm on random networks with embedded communities and on real-world networks with known hierarchical structures and use our partition similarity measure to score the correctness of our results. Our conclusions are presented in Section 5.

2. The sparsest cut

In unweighted graphs where the weight (or capacity) of each edge is assigned a unit value, the density of an \((S, T)\)-cut is \(|S \cup T|/|S \cap T|\) where \(|S \cup T|\) is the number of edges between communities \(S\) and \(T\) with \(|S \cap T|\) the maximum number of edges possible. The minimum density cut in the graph is the sparsest cut. When the edges are weighted, the density is the average weight of the \(|S \cap T|\) entries, with absent edges treated as edges of zero weight. The following Subsections 2.1–2.3 illustrate how successive sparsest cuts partition – and thereby hierarchically decompose – an example graph of Padgett’s Florentine families network. For their research, Breiger and Pattison (1986) selected a subgraph of 16 families from a larger data set collected by Padgett (e.g., Padgett and Ansell, 1993) on prominent 15th century Florentine families related by marriage and business ties. We have omitted the one disconnected family to form a 15-vertex connected graph for our example. The network is illustrated in Fig. 1. In Subsection 2.4 we provide the foundations for this hierarchical decomposition based on minimum density cuts.

2.1. The first example cut

The sparsest cut in the unweighted example graph illustrated in Fig. 1 is the edge \((9, 13)\). Community \(S\) contains the vertices \((9, 13)\). The remainder of the graph consists of set \(T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14\}\).
12, 14, 15). Only one-edge is in the cut, i.e., \(|S, T| = 1\). The maximum possible number of edges between \(S\) and \(T\) with every vertex in \(S\) connected to every vertex in \(T\) would be \(|S| |T| = 2 \cdot 13 = 26\). Thus, the density of this \((S, T)\)-cut is \(|S| |T|/(|S| + |T|) = 1/26\).

Let us contrast this one-edge sparsest cut to another one-edge cut; say edge \((10, 13)\). For this example, \(S = \{10\}\) and \(T\) contains the remaining 14 vertices of the graph. The density of the edge \((10, 13)\) cut is \(|S| |T|/(|S| + |T|) = 1/(1 \cdot 14) = 1/14\).

The edge \((9, 13)\) cut at 1/26 is less dense than the edge \((10, 13)\) cut at 1/14. So the edge \((9, 13)\) cut is the sparser cut, and indeed, it is the sparsest cut in this example graph.

2.2. The second example cut

The sparsest cut in the largest remaining component of the graph consists of the three edges \(((3, 9), (4, 7), (12, 14))\) separating the five vertices of \(S = \{3, 4, 5, 11, 14\}\) from the eight vertices of \(T = \{1, 2, 6, 7, 8, 9, 12, 15\}\). The density of this cut is 3/58 = 3/40.

Let us consider a different five-vertex to eight-vertex cut. Let \(S = \{4, 5, 11, 12, 14\}\) and \(T = \{1, 2, 3, 6, 7, 8, 9, 15\}\). We have just switched vertices 3 and 12 into the opposite vertex sets. With this example, the sizes of the vertex sets \(S\) and \(T\) remain the same at five and eight, respectively. However, the cut set of edges now contains four edges instead of three. They are \(((3, 5), (4, 7), (9, 12), (12, 15))\). The density of this cut is 4/58 = 4/40 which is more dense than the sparsest cut of 3/40 given above.

2.3. The sparsest-cut hierarchy

Continuing to find the sparsest cuts in each remaining “denser component” in the example graph of Fig. 1, we obtain a sequence of successively higher minimum density cuts. Employing the densities as measures of average linkage between groups, the results may be represented by the “divisive average linkage” dendrogram (e.g., Matula, 1986) shown in Fig. 2. Here the sparsest-cut densities range from 1/26 = 0.03846 up to the four pairs and one triple identified as terminal cliques in this example hierarchy.

2.4. Uniqueness of the sparsest-cut hierarchy

It is always possible to obtain a hierarchical decomposition into individual actors (vertices) by iteratively continuing to determine sparsest cuts to successively partition each remaining component. When any such component has more than one sparsest cut (i.e., cuts of the same minimum average density), we remove all edges of all sparsest cuts forming a multipartite partition at that iteration in the hierarchical decomposition. We shall now show in general that when the sparsest cut of each successive component is unique, the minimum cut density levels will monotonically increase yielding a bifurcating dendrogram. The scale of the dendrogram is provided by the minimum cut density levels, as illustrated in Fig. 2.

In the following discussion it is convenient to represent the graph \(G\) with a symmetric weighted adjacency matrix \(A\). If there is an edge between vertices \(i\) and \(j\) in \(G\), then the entries \(a_{ij}\) and \(a_{ji}\) of \(A\) are equal to the weight of the edge \((i, j)\); otherwise, \(a_{ij} = a_{ji} = 0\). If the graph is unweighted, then \(a_{ij} = a_{ji} = 1\) for every edge \((i, j)\). The successive components resulting from the sequence of unique sparsest cuts are now shown in general to each have higher density than the sparsest cut that determined these components. The density of graph \(G\), or any generated component of \(G\), is given by the average of the upper-diagonal entries of \(A\). The density of any sparsest cut \((S, T)\) is given by the upper, off-diagonal \(|S| \cdot |T|\) rectangular region of the reordered adjacency matrix \(A\).

Lemma 1. Let \((S, T)\) be a sparsest cut of the graph \(G\). Then the density of the cut is less than or equal to the density of \(G\).

Proof. The smallest average weight of any row (excluding diagonal elements) in \(A\) is less than or equal to the density of \(G\). Since the average weight of a row is the density of the cut separating that vertex from the remaining graph, this row average is then greater than or equal to the minimum density obtained for the sparsest cut.

Lemma 2. Let \((S, S')\) be a unique sparsest cut of \(G\), and let \((T, U)\) be a sparsest cut of the component determined by \(S'\), i.e., the induced subgraph \((S')\). Then the density of cut \((T, U)\) in \((S')\) is greater than the density of cut \((S, S')\) in \(G\).

Proof. Let \(\sigma_{ST}\) be the sum of the weights on the edges between \(S\) and \(T\), with \(\sigma_{SU}\) and \(\sigma_{TU}\) defined similarly. Assume without loss of generality that

\[
\frac{\sigma_{ST}}{|S| \cdot |T|} \leq \frac{\sigma_{SU}}{|S| \cdot |U|}.
\]

In the following we use the fact that \(A/B\) strictly less than \(C/D\) implies that \(A/B\) is strictly less than \((A + C)/(B + D)\) which in turn is strictly less than \(C/D\), where \((A + C)/(B + D)\) is called the mediant (e.g., Hardy and Wright, 1979). It follows from (1) by taking the mediant that

\[
\frac{\sigma_{ST}}{|S| \cdot |T|} \leq \frac{\sigma_{ST} + \sigma_{SU}}{|S| \cdot (|T| + |U|)}.
\]

Observe that \(S = T \cup U\), and so the right-hand side of (2) is the density of the unique sparsest cut \((S, S')\) of \(G\). Now, suppose that the density of the sparsest cut in \((S')\) is less than or equal to the density of the sparsest cut in \(G\). That is, suppose that

\[
\frac{\sigma_{TU}}{|T| \cdot |U|} \leq \frac{\sigma_{ST} + \sigma_{SU}}{|S| \cdot (|T| + |U|)}.
\]

Then the mediant of the left-hand sides of Eqs. (2) and (3) must also be less than or equal to the common right-hand side bound of (2) and (3) yielding

\[
\frac{\sigma_{ST} + \sigma_{TU}}{|T| \cdot (|S| + |U|)} \leq \frac{\sigma_{ST} + \sigma_{SU}}{|S| \cdot (|T| + |U|)}.
\]

Observe, however, the expression on the left-hand side of (4) is the density of the cut \((T, S \cup U)\) in \(G\). Thus, (4) contradicts the statement that \((S, S')\) is the unique sparsest cut of \(G\). Therefore, the density of the sparsest cut \((T, U)\) of \((S')\) must be greater than the density of the unique sparsest cut of \(G\).

Lemmas 1 and 2 establish the following result characterizing a unique divisive average-linkage dendrogram.
Theorem 3. Let the $n$-vertex weighted graph $G$ have a sequence of $(n-1)$ unique sparsest cuts of successive components where each unique sparsest-cut partitions a component into two smaller components, until each component contains a single vertex. Then the hierarchical decomposition forms a dendrogram with the partition levels given by the density of the sparsest cuts.

When there is more than one sparsest cut at a given level for a particular weighted graph, it is possible, but not guaranteed, that all parts of the multipartite cut determined at that level will have higher density sparsest cuts. Such cases may be investigated and continued as long as the hierarchy yields monotonically increasing sparsest-cut densities. This is the case for the example of Fig. 2.

In the next section we describe how the maximum concurrent flow problem may be solved by a linear program and tailored to build the sparsest-cut dendrogram as long as successive sparsest cuts are sufficiently sparse to be the constraining bound on the concurrent flow at each level.

3. The MCF cut community structure algorithm

The main step of our community structure algorithm is to find the sparsest cuts in a graph, or in a connected component of a graph. This step is repeated to cut the largest (or least dense) remaining component until some stopping criterion such as Newman’s (2004) highest weighted modularity $Q^*$ value or the greatest average density of neighbor vertices is reached. The sparsest-cut problem is NP-hard (Matula and Shahrokhi, 1990), and so it is unlikely that it can be solved in all cases with an efficient (polynomial-time) algorithm. However, the sparsest-cut problem is closely related to the MCFP which can be solved efficiently via linear programming. The MCFP extends the Ford and Fulkerson (1956, 1962) source–sink max-flow problem to all-pairs maximum concurrent flow. A famous result from Ford and Fulkerson (1956, 1962), the max-flow–min-cut theorem, is that the maximum flow between a given source vertex $s$ and sink vertex $t$ in a graph is equal to the capacity of a minimum capacity cut separating $s$ and $t$. Similarly, it can be shown that the value of the maximum concurrent flow (MCFP) in a graph is less than or equal to the density of a sparsest cut. More significantly, when the MCFP solution identifies a partition into two to four parts, a sparsest cut can be identified (Matula, 1985). Similarly, it can be shown that the two to four parts are precisely the components obtained by removing the edges of all sparsest cuts. This section describes the MCFP and how our algorithm uses its solution to find sparsest cuts. Since our algorithm solves the MCFP to find sparsest cuts, and these cuts are in turn used to decompose the graph into communities, we call it the MCF cut community structure algorithm.

3.1. Description of the MCFP

The MCFP (e.g., Matula, 1985; Biswas and Mathula, 1986; Shahrokhi, 1987, 1989; Shahrokhi and Matula, 1987; Shahrokhi and Mathula, 1990; Thompson, 1985) is a multimmodity flow problem in which every pair of vertices can send and receive flow concurrently. The term throughput is defined to be the ratio of the flow supplied between a pair of vertices to the predefined demand for that pair. The throughput must be the same for all-pairs of vertices for a concurrent flow. This problem can be formulated as a linear programming problem. The objective of the MCFP is to maximize the throughput, subject to fixed capacity constraints on the edges.

3.2. Formulation of the MCF problem by linear programming

The MCFP can be formulated as a straightforward linear programming (LP) problem that works with the set of all possible paths in the graph to find the flow on individual paths so that the throughput is maximized equally between every pair of vertices in the graph. The following discussion assumes a familiarity with LP. For a broad introductory treatment on LP theory and applications see Winston and Venkaramann (2003).

3.2.1. Variables and terminology

A path $p$ in graph $G=(V,E)$ is a sequence of alternating vertices $v \in V$ and edges $e \in E$ where none of the vertices are repeated. The flow on a path $p$ is represented as $f_p$. The demand for flow for a vertex pair $(i,j)$ is represented as $d_{ij}$. $P_e$ denotes the set of all paths with endpoints $i$ and $j$. We define the set of all paths $P$ as the union of all sets $P_e$ for all vertex pairs $i,j \in V$ where $i < j$. Note that we assume, without loss of generality, that the vertices are numbered $1,2,\ldots,n$ where $n$ is equal to $|V|$.

The set of all paths that use edge $e \in E$ is represented as $P_e$. The capacity for flow on edge $e$ is represented as $c_e$.

The throughput for vertex pair $(i,j)$, $Z_{ij}$, is the ratio of the flow supplied between a pair of vertices to the predefined demand for that pair which can be written as:

\[ Z_{ij} = \sum_{p \in P_e} \frac{f_p}{d_{ij}} \]  

(5)

3.2.2. Constraints

The combined flows of all paths using a particular edge must not exceed the capacity of that edge $c_e$.

\[ \sum_{p \in P_e} f_p \leq c_e \quad \forall \ e \in E \]  

(6)

The flow on any path must be non-negative.

\[ f_p \geq 0 \quad \forall \ p \in P \]  

(7)

The throughput $Z$ is required to be equal between every pair of vertices in the graph.

\[ Z = Z_{ij} \quad \forall \ i \in V, \ j \in V, \ i < j \]  

(8)

3.2.3. Objective

The objective is to maximize the throughput.

Maximize $Z$

(9)

Maximizing $Z$ subject to (5)–(8) is a linear program.

3.3. Determining the sparsest cut from the MCFP

The MCF cut community structure algorithm solves the LP corresponding to the MCFP in graph $G$ using the weight of edge $e$ as $c_e$ and the demand $d_{ij} = 1$ for all distinct vertex pairs. The resulting throughput is called $Z(1)$ and corresponds to the throughput at cut level 1.

There can be multiple MCFP solutions for a given graph. However, it can be shown (e.g., Matula, 1985) that there is a unique set of critical edges $F$ that are saturated by every optimal assignment of flow values $[f_p | p \in P]$, so that

\[ \sum_{p \in P_e} f_p = c_e \quad \forall \ e \in F \]  

(10)

It can be further shown (e.g., Matula, 1985) that removing the critical edges partitions the graph into two or more components. Furthermore, the remaining components are composed entirely of non-critical edges which, therefore, allow a higher level of throughput between all vertices of each component. The fact that the critical edges are exactly the edges between the two or more distinct components of the resulting partitions is a fundamental property of the...
MCF problem. For example, in Fig. 1 the maximum throughput is $Z(1) = 1/26$ and $F$ consists of the single edge $(9, 13)$. Observe that the throughput is constrained to $1/26$ since edge $(9, 13)$ must carry equal amounts of flow between 26 demand pairs in any concurrent flow. The maximum throughput in the largest component remaining after edge $(9, 13)$ has been removed from the graph in Fig. 1 is $Z(2) = 3/40$ with $F = (3, 9), (4, 7), (12, 14)$. In this case flow between 40 demand pairs must cross the three edges in $F$ and no other cut in the component is more constraining on the throughput.

We find $F$ with an iterative perturbation process. For example, to find the first set of critical edges, we take each edge $e$ saturated by the MCF, let $c_e = c_e - \epsilon$, where $\epsilon$ is a small number relative to $c_e$, and then solve the MCFP and compare the resulting throughput $Z$ to $Z(1)$. If reducing $c_e$ reduces the throughput ($Z < Z(1)$), then $e$ is critical (i.e., $e \in F$). On the other hand, if the throughput with the reduced $c_e$ is still $Z(1)$, then $e$ is not critical.

It is important to point out that for some graphs the density of a sparsest cut is strictly greater than the maximum throughput (Matula, 1985), and that in general the set of critical edges $F$ is not necessarily a sparsest cut in the graph. Therefore, the procedure described above is technically just a heuristic rather than an exact algorithm for finding a sparsest cut. It is an exact algorithm for finding the set of critical edges that partition a component into two or more parts. However, it can be shown that whenever removing the critical edges leaves a partition of just two to four components, $F$ contains a subset of edges $F$ that is a sparsest cut (Matula, 1985).

Furthermore, if $F$ partitions the graph into two components, as was often the case in our experiments, then $F = F$. It is also important to point out that a graph may have multiple sparsest cuts. If the decomposition yields a component that has more than one sparsest cut, then removing all edges of all sparsest cuts at that density level would yield a uniquely determined multipartite partition in the hierarchical decomposition. It should be noted that such a multipartite cut could in general generate a component having a smaller density. However, it can be shown that when the throughput equals the sparsest-cut density, any such multipartite partition must have all resulting components having larger minimum density cuts preserving the hierarchical generation of the dendrogram. If the MCFP solution generates a multipartite partition without identifying a sparsest cut, then we terminate the dendrogram of that component noting the component is sufficiently homogenous to justify no further subdivision. With this stopping condition, the maximum concurrent flow solutions iteratively establish the divisive average-linkage dendrogram.

3.4. Main steps of the MCF cut community structure algorithm

The procedure described above is applied iteratively until the community structure of the graph (i.e., social network) is identified. The MCF cut community structure algorithm can be described at a high-level as consisting of the main steps listed below. Note that if the social network is a connected graph, then the whole network is treated as the initial, selected component.

DO UNTIL stopping criteria are reached
| Select a component to cut. |
| Solve the MCFP for the selected component. |
| Identify the critical edges. |
| Remove critical edges to produce new components. |
END DO UNTIL

The component selection may be based upon a variety of criteria. The selected component may be the largest remaining component if the sizes of the anticipated groups are approximately the same. Or, the least-dense component may be selected when the sizes of the groups are expected to vary widely. A dense component is likely to contain a high-affinity community already. A component that is less dense may contain multiple communities connected by lower-capacity inter-community edges causing a bottleneck in the flow between communities.

3.5. Practical considerations for implementation

The LP formulation for MCFP given above is not efficient in the sense that it explicitly represents all paths in $G$, and the number of paths can be quite large with respect to the number of vertices. The MCF problem can be reformulated to provide a more tractable linear program (Ahuja et al., 1993; Shahrokhi and Mathula, 1990), establishing that the problem is in the class P of problems having polynomial-bounded solutions. But, that subject is beyond the scope of this paper. The results presented here were obtained by more efficient LP formulations using commercially available LP packages. Even with the more tractable formulations, however, the perturbation process described above can be quite time consuming. As an alternative, we can apply linear programming duality theory to determine a cut of density $Z$ directly from the MCF solution. Along with the optimal values for the flow variables, the LP solution gives a so-called shadow price, or dual cost, for each edge (Ahuja et al., 1993). The shadow price of an edge $e$ can be interpreted as the rate at which the throughput changes as a function of the capacity $c_e$. Given an MCF solution, let $\hat{E}$ denote the set of saturated edges with non-zero shadow prices. It can be shown that an edge in $\hat{E}$ must be critical, and that the set of edges in $\hat{E}$ form a cut with density $Z$. It should be noted, however, that this method may only identify one cut in cases where there are multiple sparsest cuts, or, more generally, when there are ties for the most constraining cut on $Z$. Thus, the last step of the loop in Section 3.4 considers three cases:

1. If the MCF saturates all edges in the given component, then for our purposes the component is sufficiently dense to say that it has no meaningful subcommunities and we partition it into individual vertices.
2. If removing $\hat{E}$ from the component creates a multipartite partition with more than four components, then for our purposes the component is sufficiently dense to say that it has no meaningful subcommunities and we partition it into individual vertices.
3. If neither of the other two cases holds, then we remove $\hat{E}$ and partition the component into two, three, or four parts (subcommunities).

4. Experimental results

In this section, we report the application of the MCF cut algorithm to a real-world network with a given community structure to evaluate how effectively the algorithm can identify the known hierarchical community structure. Then we discuss the experiment with a series of randomly generated weighted networks that have embedded community structures and show how the accuracy of community structure algorithms can be evaluated by counting the number of edges each algorithm correctly identifies as inter-community and intra-community types.

4.1. Professional football network

A real-world network with a known hierarchical structure is the seasonal schedule of games between teams of the National Football League (NFL) of American professional football.1

---

1 The schedule history of the NFL games discussed in this paper is available via the Internet at http://www.nfl.com/teams/schedule.
4.1. Hierarchy of American professional football

The NFL consists of the American Football Conference (AFC) and the National Football Conference (NFC). Each of these two conferences contains four divisions (East, North, South, and West) with four teams in each division.

A team plays the other three teams in its division twice per regular season so that one game is held on its own home field and the other game is played on the opponent’s home field. A team also plays all four teams in a different division within its conference. From each of the remaining two divisions within its conference, the team plays only one of the teams. The particular teams selected for these individual inter-division games are based on prior years’ win–loss records. This allows that two of the scheduled matches are effectively random choices compared to all others satisfying a fixed hierarchical plan. This makes 12 games played within the team’s conference per season. In the other conference, all four teams in a particular division are played and none of the teams in the other three divisions of the other conference are played. So, the four games from the other conference plus the 12 games within the conference makes a total of 16 games played per team during the regular season. The divisions playing each other are rotated each year so that one game is held on its own home field and the other game is played on the opponent’s home field. The Girvan–Newman algorithm was not able to correctly identify this hierarchical plan within the NFL conference networks. The order in which the edges with the highest edge betweenness centrality value, are removed, does not identify the correct members of all eight NFL divisions by iteratively removing the edges with the highest edge betweenness centrality value, the network hierarchy is not identified. The order in which the divisions are separated from the network is not indicative of the community structure algorithms, the regular-season schedule ensures that teams within the same division play each other twice each year, but teams outside the division in a different conference may not play each other but once in several years.

4.1.1. Results of the MCF cut algorithm

The MCF cut algorithm found the first sparsest cut at a density of 0.75 which evenly divided the NFL into the AFC and NFC conferences of 16 teams apiece as shown in Fig. 5 by the dotted lines. The next cut at density 1.50 partitioned all four divisions of the AFC simultaneously and all four divisions of the NFC. This is consistent with the regular pattern of games between conferences and divisions. The final sparsest cuts at density 6.00 partitioned the network into the 32 individual teams.

The MCF cut algorithm thus yields the dendrogram of Fig. 5, where the cut densities provide the scale for the hierarchical decomposition. The hierarchy of dense, weighted networks like this is revealed by the sparsest cuts from the MCF cut algorithm. This is beneficial for the analysis of social networks that may have subgroups within groups of individuals.

Although the weighted Girvan–Newman (2004) algorithm finds the correct members of all eight NFL divisions by iteratively removing the edges with the highest edge betweenness centrality value, the network hierarchy is not identified. The order in which the divisions are separated from the network is not indicative of the conference they belong to. The Girvan–Newman algorithm was not Q1
Fig. 4. Adjacency matrix of NFL 2004–2006 seasons showing hierarchical clustering by the min density cuts.

Fig. 5. Dendrogram showing hierarchy of professional American football.

4.3. Measure of community structure similarity

We have defined a similarity measurement that allows comparisons of any two partitions of a network without the necessity of mapping parts of each partition into parts of other partitions. Given a particular partitioning, every edge in a community structure network \( G = (V, E) \) can be classified as either of two types. An edge is either a member of the inter-community type edge set \( CB \) (correct edges between two different communities), or it is a member of the intra-community type edge set \( CW \) such that \( E = CB \cup CW \). A divisive community structure algorithm identifies inter-community type edges as the set \( LB \) (algorithm-labeled edges between two different communities). The remaining edges of the network \( E - LB \) would be what the algorithm left as the set of intra-community edges.

4.2. Community structure fundamentals

Communities within a network contain elements that have a greater affinity to the other elements inside the community they belong to than to elements outside their community. This affinity is reflected in the number and strength of their links to each other. The links in a network represented as edges in a graph consist of two types—the intra-community edges binding together the members within their communities and the inter-community edges between members of two different communities.

The ideal divisive community structure algorithm would be able to identify all of the inter-community edges for removal without misidentifying any of the intra-community edges as being inter-community edges. When the inter-community edges are removed from the network, all that remains are the disconnected communities with the intra-community edges tying their member vertices together. This observation provides the standard for edge type identification against which the similarity of network partitionings from community structure algorithms can be compared.
In comparing the similarity of two partitionings of a network, one is designated the benchmark against which the other is measured. The extent to which the measured partitioning conforms to the benchmark is seen in how closely the two types of identified edges match those of the benchmark expressed as a fraction. The numerator of this fraction is the number of the inter-community type edges plus the number of the intra-community type edges correctly identified by the algorithm. The denominator is the total number of edges of both types \( |E| \) actually in the network.

The measure of accuracy, or score, is defined to be

\[
M = \frac{|L_G \cap C_W| + |(E - L_G) \cap C_W|}{|E|} \quad (11)
\]

In our experiments, the benchmark networks described in Section 4.5.1 below were generated with a known partitioning. The measure of accuracy \( M \) was used to evaluate the partitioning found by both of the community structure algorithms tested. A score of 1.000 indicates that the algorithm correctly identified the inter- and intra-community type of every edge in the network correctly. Lower scores reflect the percentage of edges that were correctly typed.

### 4.4. Collegiate football network

The organization of football conferences of the National Collegiate Athletic Association (NCAA) provides another example of a real-world, hierarchical-structured network. The particular example presented here is comprised of the 120 teams of the NCAA Football Bowl Subdivision\(^2\) formerly designated, prior to 2006, as the NCAA Division I-A. The teams competing in this network are from the American colleges, universities, and military academies playing American football. The edges of our example are weighted with the number of games each pair of teams played against each other over the 3-year period of 2005–2007. These weightings, which can range from 0 to 3, are an indication of attraction or similarity.

#### 4.4.1. Hierarchy of the NCAA Football Bowl Subdivision

The NCAA Football Bowl Subdivision over the 2005–2007 years consisted of 11 conferences and 4 independent teams who are not conference members. Five of the conferences are further subdivided into a pair of divisions. This makes for a total of 20 communities in this network ranging in size from 1 to 11 teams. Two of the non-subdivided conferences have 8 teams, two have 9 teams, and the other two conferences have 10 and 11. Four of the subdivided conferences contain twelve teams separated into six teams in each of their divisions. One conference is asymmetrically organized with one division of six teams and a second division of seven teams. With a large network of so many communities of different sizes at different hierarchy levels, this real-world network presents a formidable challenge to any community structure algorithm.

Generally, teams play the other teams in their own conference more often than they play teams outside of their conference. So the weightings on the edges indicating games played are a measure of similarity. Although some teams play rivals in other conferences every year, the weights on the inter-community edges tend to be lower than the weights on the intra-community edges.

The NCAA recognizes the formal, 15-part partitioning of the Football Bowl Subdivision into four independent teams and eleven conferences with their divisions below them forming a 20-part partitioning. But, there is no classification by the NCAA for any intermediate hierarchy between the 120-team network as a whole and the 15-part partitioning. Any hierarchy determined from the data between the whole network and the 15 communities is subject to further social network analysis. The results revealed by the MCF cut algorithm show a clear regional and demographic interpretation for this portion of the network hierarchy.

This 120-vertex collegiate football network contains 1012 weighted edges representing games played between pairs of teams over the 3-year period. The games played within the various conferences and divisions (i.e., the communities) classify 383 of the edges as intra-community edges. The remaining 629 edges are the inter-community edges representing games played between teams from different communities. The degree of a vertex is the number of distinct opponents it played over the 3-year period. The average degree is 16.87. Temple University, playing as an independent in 2005 and 2006 before joining the Mid-American Conference (MAC) in 2007, had the maximum degree of 23. Western Kentucky University had the minimum degree of 9 since it has only been in the NCAA Football Bowl Subdivision for the 2007 season when it moved from the NCAA Football Championship Subdivision (formerly designated the NCAA Division I-AA). Western Kentucky played as one of the four independent teams with the United States Military Academy (Army), United States Naval Academy (Navy), and Notre Dame making up the other three.

The eccentricity of a vertex is the longest distance of all the shortest paths from that vertex to any other vertex in the network. Every vertex in this network has an eccentricity of 3. So the diameter and the radius of the network are both 3. This means that every team is within three matches of playing each other. For example, if team \( s \) and team \( t \) are two of the teams at greatest distance from each other; then team \( s \) has played a team \( v \) and team \( r \) has played a team \( u \) that have played each other such that the path between \( s \) and \( t \) is \( s\rightarrow v\rightarrow u\rightarrow t \), a distance of three hops.

Both the weighted Girvan–Newman algorithm and the MCF cut algorithm were employed to analyze this network. The measure of accuracy of Eq. (11) serves to indicate the accuracy of these two algorithms.

#### 4.4.2. Results of the weighted Girvan–Newman algorithm

Both the weighted Girvan–Newman algorithm and the MCF cut algorithm found the same first cut separating off the 28-team component comprised of the western USA conferences of the Mountain West Conference (MWC), the Western Athletic Conference (WAC), and the Pacific-10 Conference (PAC 10) from the remaining 92 teams. However, the MCF cut algorithm efficiently removed 94 of the inter-community edges while the weighted Girvan–Newman algorithm removed an additional 14 edges that belonged to other cuts that would not be fully partitioned until later in the run. Since those 14 extra edges were inter-community edges between other communities, the algorithm was not penalized by the accuracy score because it is not dependent upon how the final results are derived.

The accuracy score for the weighted Girvan–Newman algorithm peaked at 0.985. At this point the algorithm had removed enough edges to break the network into the 20 communities that actually comprise the real network. The algorithm had correctly identified all 629 inter-community edges, but it had removed an additional 15 intra-community edges that should not have been removed. Otherwise, the weighted Girvan–Newman algorithm correctly found the hierarchy of the NCAA football conferences and divisions.

#### 4.4.3. Results of the MCF cut algorithm

The MCF cut algorithm achieved a perfect 1.000 measure of accuracy score at its peak correctly identifying the NCAA football conferences and divisions of all 20 communities. The algorithm had

---

\(^2\) The NCAA Football Bowl Subdivision network discussed in this paper was created from the 2005–2007 schedule history available via the Internet at [http://www.ncaa.org/stats/football/footballMenu.html](http://www.ncaa.org/stats/football/footballMenu.html) under the Weekly Game Summaries of Division I Bowl using the first 13 weeks of play for each year.
Fig. 6. Dendrogram of NCAA Bowl Division Football 2005–2007 seasons from MCF cut algorithm. The authors thank for preparing the illustrations in this figure and Fig. 7 to represent our NCAA Football Bowl Subdivision network.

correctly identified all 629 of the inter-community edges without mistaking any of the intra-community edges.

The dendrogram determined by the MCF cut algorithm for the NCAA football 2005–2007 seasons data is shown in Fig. 6. The vertical scale from 0 down to 1.5 indicates the densities (average weights) of the edges in the successive sparsest cuts determined by the solutions to the 19 MCFPs. All solutions were unique sparsest cuts, which by Lemma 2, guarantees the monotonically increasing levels of the cut density as we traverse down the dendrogram. Effectively for this data, the MCF cut algorithm generated the divisive average-linkage dendrogram through this 20-community partition. The width of the vertical lines in the dendrogram is drawn proportionally to the densities of the resulting components, which increase after each set of sparsest-cut edges is removed.

The weighted adjacency matrix is plotted in Fig. 7a with the 120 teams ordered alphabetically, yielding a reasonably random plot. The gray scale of the individual entries indicates whether the particular pair of teams played three, two, one or no games between each other in the 3 years. The plot of Fig. 7b illustrates the blocked structure of the adjacency matrix after the 14 sparsest cuts through density 1.0 are determined. Within each diagonal block, the teams are still displayed in alphabetical order and are subsequently resolved into denser blocks yielding the divisions of the five conferences with divisions as noted in the dendrogram at sparsest-cut level 1.5. The five sparsest cuts are seen as sparse, off-diagonal regions successively labeled 1–5 in Fig. 7b showing how extracting these sparse regions yields denser blocks on the main diagonal of the resulting block-structured matrix.

It is important to note that there are numerous entries of the maximum weight of three in even the sparsest-cut regions labeled 1–5 in Fig. 7b. This predisposes typical agglomerative algorithms such as single-, complete-, and even average-linkage “bottom up” procedures to have non-deterministic starting pairings and possibly lead to irreparable misleading cluster hierarchies. This data set shows the robust nature of the MCF cut algorithm in determining a complex hierarchy of variable-size communities.

4.5. Randomly generated weighted networks

The experiments with randomly generated, weighted networks were designed to explore the accuracy of the MCF cut algorithm over an increasing range of densities as the average degree of the 32-vertex networks was varied from 8 to 31. The weighted Girvan–Newman algorithm was run against the same set of networks.

4.5.1. Creating the networks

A benchmark set of weighted networks were randomly generated to compare the accuracy of the MCF cut algorithm to the weighted Girvan–Newman algorithm. The benchmark networks contained 32 vertices with 4 embedded communities of 8 vertices apiece. Each community was a complete subgraph. Each vertex was connected to its seven neighbors within its community by a weighted edge so that every community contained 28 intra-community edges. With four communities in the network, a network would have a total of 112 intra-community edges. Every vertex in the network had a degree of at least seven because that is the number of intra-community edges incident upon it.

The inter-community edges were added randomly to pairs of vertices in different communities to increase the average degree of the network one degree at a time to form the test series. So a network with an average degree of eight would have 112 intra-community edges and 16 inter-community edges. For an average degree of 9, there are 112 intra-community edges and 32 inter-community edges, and so on until the network itself becomes a complete graph with every vertex at degree 31 (i.e., an edge connects it to all 31 of its neighbors). That network contains 496 edges—112 intra-community edges and 384 inter-community edges.

The weights assigned to the edges were sampled from an infinite population of normally distributed (i.e., Gaussian) variates with a mean of 2.0 for the intra-community edges and a mean of 1.0 for the inter-community edges. Only the non-negative values from the distribution were used for the weighting. The standard deviation of the intra-community edge weights was 0.6667 and the standard deviation of the inter-community edge weights was 0.3333 so that the tails of the two bell-shaped distribution curves shown in Fig. 8 overlapped. This ensured that on average the edges within the communities had a higher weighting than the edges between the communities. But, there were occurrences where some inter-community edges were assigned higher-valued weights than some intra-community edges.

Please cite this article in press as: Mann, C.F., et al., The use of sparsest cuts to reveal the hierarchical community structure of social networks, Soc Netw (2008), doi:10.1016/j.socnet.2008.03.004
4.5.2. Evaluating the algorithms

Both the MCF cut algorithm and the weighted Girvan–Newman algorithm were run on the same set of randomly generated weighted networks. Twenty networks were generated with the results averaged for each point appearing in Figs. 9 and 11 plots. Using the measure of algorithm accuracy (11) defined in Section 4.3 above, the scores for the two algorithms are shown plotted in Fig. 9 as a function of the average degree. Each algorithm produces a series of partitions, and the values reported in Figs. 9 and 10 are the scores and communities produced by the partitions that gave the highest modularity $Q^w$ value for weighted networks (e.g., Newman, 2004). The MCF cut algorithm retains an almost perfect score over the range. The Girvan–Newman algorithm decreases in accuracy as the average degree of the networks increase.

A high score indicates that almost all of the inter-community edges are being correctly identified. Consistent with this, is an accurate identification of the four embedded communities. Fig. 10 shows that the MCF cut algorithm maintains a community-identification count very close to four with its high scores. The data for Figs. 9 and 10 plots are given in Table 1.

As the average degree increases, the edge betweenness centrality values found by the Girvan–Newman algorithm leads it to remove more of the intra-community edges instead of inter-

![Fig. 7. (a) Adjacency matrix of collegiate football network arranged alphabetically. (b) Adjacency matrix of collegiate football network after cuts through density 1.0.](attachment:image.png)
Fig. 8. Distributions of edge weights.

Fig. 9. Accuracy of edge types correctly identified.

Fig. 10. Number of communities found by average degree.

5. Size considerations

The largest networks that can be handled by the MCF cut algorithm are currently limited by the linear programming implementation to those around the size of the 120-vertex collegiate football network. Although it may be possible to improve the LP problem approach, the MCF cut algorithm is near the practical size limit for this method. However, the MCFP flow-converging algorithms developed by Biswas and Mathula (1986) should allow the MCF cut algorithm to solve community structure network problems at least one order of magnitude larger and perhaps two orders of magnitude larger than the LP implementation of the algorithm (e.g., Shahrokhi and Matula, 1987). We intend to complete the development of a non-LP flow-converging version of the MCF cut algorithm to explore larger networks in future research.

6. Conclusion

Our research shows that the MCF cut algorithm provides more accurate results than the weighted Girvan–Newman algorithm in finding the community structure of dense, weighted social networks. This algorithm, based upon the maximum concurrent flow, finds the divisive average-linkage hierarchy through the density of the sparsest cuts. The density levels identify the hierarchy. We have determined conditions that assure that the densities increase monotonically with each cut providing a hierarchy that can be represented in a dendrogram scaled by the cut densities. This MCF cut algorithm can serve as a tool to accurately reveal the hierarchical community structure of highly dense, weighted networks.

References


Please cite this article in press as: Mann, C.F., et al., The use of sparsest cuts to reveal the hierarchical community structure of social networks. Soc Netw (2008). doi:10.1016/j.socnet.2008.03.004