

Trusses: Cohesive Subgraphs for Social Network Analysis

Jonathan Cohen
National Security Agency
Suite 6514
9800 Savage Road
Fort Meade, MD 20755-6513

Abstract

Social network analysts employ automated methods of identifying cohesive subgraphs as a means of focusing their attention on areas of the network that are likely to be fruitful. A variety of standard cohesive subgraphs have been defined in the literature, but most suffer from computational intractability and other drawbacks.

This paper introduces a new cohesive subgraph called a truss. Like most of the existing definitions, the truss is a relaxation of a clique. The truss offers guaranteed computation in polynomial time, is motivated by a natural observation of social cohesion, and is related nicely to other standard structures.

After a review of other cohesive group definitions, the truss is defined, many of its properties are enumerated, a detailed accounting of computing trusses is given, and some examples are shown.

Table of Contents

<i>Introduction</i>	3
<i>The Truss</i>	6
<i>Motivation and definition</i>	6
<i>Properties of trusses</i>	8
<i>Finding Maximal Trusses</i>	16
<i>Examples</i>	19
<i>Krackhardt's high-tech managers</i>	19
<i>Freeman's EIES researchers</i>	20
<i>Magazine article similarity graph</i>	23
<i>Concluding Remarks</i>	27
<i>Acknowledgements</i>	27
<i>References</i>	28

Introduction

Graphs are frequently used to capture and study patterns of ties between social actors. Actors are represented by graph vertices; edges between the vertices represent relationships among the corresponding actors. The simple pictorial nature of graphs goes a long way toward human comprehension of the network of relationships, and the structural nature of graphs provides easy manipulation of the data.

Many of the operations performed on social network graphs seek to identify cohesive subgroups of actors, represented by subgraphs within the larger network, for further human analysis.

Moody observes that while most of the methods of social network analysis were developed for necessarily-small graphs, social analysis is now attempting to address networks of rapidly-increasing size (Moody 2001). Accordingly, it is important to identify methods that are not only effective, but efficient.

The most obvious cohesive subgraph, introduced to social network analysis by Luce and Perry (1949), is the clique — a subgraph in which every vertex is adjacent to every other vertex. Usually, the definition further specifies that the clique be maximal, so that no other vertices may be added to the subgraph without it ceasing to be a clique. The clique turns out to be of disappointing utility because it is both too rare and too common: cliques of only a few members are frequently too numerous to be helpful¹, and larger cliques are too limiting to be expected even among tightly-knit actors. In addition, the automated finding of cliques is intractable, that is, the work scales worse than any polynomial of the problem size, making the enumeration of cliques impractical for moderate data sizes (Bron and Kerbosch 1973).

One may choose to generalize and relax the clique to avoid the rarity problem. This process can take several different roads, depending on which properties are maintained and which are loosened.

One property of a clique is that each member is a distance 1 away from every other member, that is, one can get between any pair of members by traversing only a single edge. The so-called n -clique (Luce 1950) is a generalization of the clique, and consists of a group of vertices such that each pair of members are separated by no more than a distance n . Note that a clique is a 1-clique. The n -clique suffers from quite a few problems: For $n > 2$, the group is likely to be too diffuse to be of interest. Also, the problem of enumerating n -cliques is no better than enumerating cliques. Further, finding n -cliques remains intractable. Finally, it may happen that intermediaries responsible for

¹ One can record statistics of joint membership in cliques and use them as a basis for clustering, but there may be more direct and far less expensive ways to achieve similar results.

proximity of an n -clique's members may not be members of the n -clique itself (Alba 1973).

Two attempts to address problems of n -cliques are n -clans and n -clubs (Alba 1973, Mokken 1979). n -clans are n -cliques whose diameter² is no bigger than n . An n -club is a maximal subgraph of diameter n . In both cases, the diameter is measured strictly within the subgraph. These modifications do not remove either the problems of enumeration (there are still too many) or of computational intractability.

Another property of a clique is that each member is adjacent to all clique members save itself. A generalization in this respect is the k -plex (Seidman and Foster 1978). A k -plex is a maximal subgraph in which each vertex is adjacent to all but at most k of the members. So a clique is a 1-plex. Again, enumeration and intractability are no better than for cliques.

On the other extreme is the simple k -core, introduced by Seidman (1983). A k -core is a maximal (single-component) subgraph in which each member is adjacent to at least k other members. It too is a generalization of the clique: a clique is a k -core with $k+1$ members. Unlike the other clique generalizations above, the k -core can be computed in polynomial time, and does not pose an enumeration problem. The disadvantage of k -cores is that they are too promiscuous: rather than being sets of high cohesion, Seidman characterizes k -cores as “seedbeds, within which cohesive subsets can precipitate out.” The k -core does narrow the search for cohesive subgroups, but at the expense of including much else.

The author sought something between the expensive-to-find and overly-numerous groupings provided by cliques, n -cliques, n -clans, n -clubs, and k -plexes on the one hand, and the inexpensive, few-in-number, but overly-generous k -cores on the other. He found it, in the form of another generalization of the clique, which he calls a truss.

The remainder of this paper is a discussion of the truss, giving its definition, noting some of its properties, describing its computation, and illustrating its application.

Before going on, it should be pointed out that in addition to the methods above, cohesive groupings may be defined on the basis of comparing internal connections to external ones. The LS-sets, borrowed from circuit partitioning for board layout (Luccio and Sami 1969), are so special as to be extremely rare in social networks and are expensive (but polynomial) to calculate (Borgatti, Everett, and Shirey 1990). A relaxation of LS-sets, λ -sets, still requires quartic work (Borgatti, Everett, and Shirey 1990). In addition, a variety of agglomerative clustering techniques have been employed, as well as divisive clustering approaches, such as Girvan and Newman's betweenness method (Girvan and Newman 2001) and Moody and White's method based on vertex

² Diameter, when an integer as opposed to a path, is the largest distance between two vertices in the graph, where distance between two vertices is defined as the minimum number of edges that must be traversed to go from one vertex to the other.

connectivity (Moody and White 2003), both of which are polynomial, but quite expensive in their computation.

The interested reader may consult the very readable book by Scott (1991) for an introduction to social network analysis. Wasserman and Faust (1994) offer a more in-depth and extensive presentation. Both discuss a variety of cohesive subgraphs and their application.

The Truss

Motivation and definition

White and Harary (2001) argue that what has been missing from many of the definitions of cohesive groups is the strength from having multiple paths joining actors for a given social relation.³ It is this point of departure that the author has used to define a “truss.” In particular, one can make this observation about social structures: if two actors are strongly tied, it is likely that they also share ties to others. We shall create something called a k -truss, in which a tie between A and B is considered legitimate only if supported by at least $k-2$ other actors who are each tied to A and to B .

The truss may be considered yet another extension of the clique. One may view the clique of order k as a subgraph in which each edge joins two vertices that have at least $k-2$ common neighbors. Stated another way, each edge is reinforced by $k-2$ pairs of edges making a triangle with the original edge. So here is the formal definition:

Definition: A k -truss is a non-trivial, one-component subgraph such that each edge is reinforced by at least $k-2$ pairs of edges making a triangle with the that edge. (Non-trivial here excludes an isolated vertex as a truss.)

Definition: A maximal k -truss is a k -truss that is not a proper subgraph of another k -truss.

Observation: A clique of order k is a k -truss.

Figure 1 offers an example. The entire graph is pictured in Figure 1(a). Subsequent images there show dimming of all but maximal k -trusses for increasing values of k . Note that there are two 3-trusses, one of which is a clique of order 3. There is one 7-truss, which is also a clique. The nice thing is that the rather close-knit subgraph shown in Figures 1(c) and (d), and with slight modification in Figure 1(e), is easily picked up.

³ White and Harary go on to suggest cohesive groups identified by vertex connectivity, a criterion that is stronger, and more expensive, than the truss.

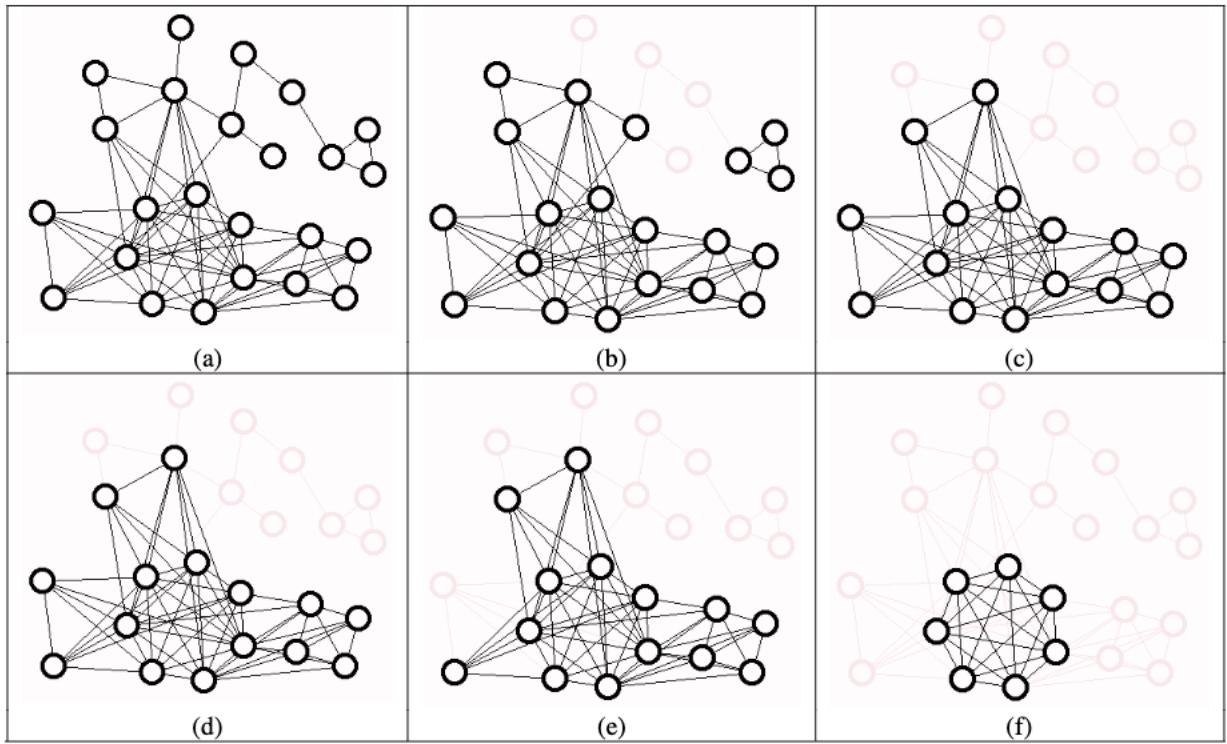


Figure 1. An Example of Maximal Trusses. (a) original graph, (b) 3-trusses, (c) 4-truss, (d) 5-truss, (e) 6-truss, (f) 7-truss. Note that (c) and (d) are the same.

Observation: Each non-trivial component of a graph is a maximal 2-truss.

Properties of trusses

Notation:

G is the graph under discussion. G is assumed undirected and simple⁴.

V is the vertex set of G .

E is the edge set of G .

n is the number of vertices in G .

$|E|$ is the number of edges in G .

$N(v)$ indicates the set of neighbors of arbitrary vertex v .

$d(v)$ denotes the degree (also called valence) of arbitrary vertex v .

Prop: Trusses can be computed in polynomial time.

P
R
O
O
F

Here is a naive, but adequate, algorithm that will compute all maximal k -trusses of graph G , each being left as a component. (See the section “Finding Maximal Trusses” for a better approach.)

```
crummy_code_to_reduce_graph_to_k_truss( Graph G )
{
    until no change do
    {
        for each edge  $e = (a,b) \in G$ ,
            if (  $|N(a) \cap N(b)| < k - 2$  ) remove  $e$  from  $G$ ;
    }
    remove isolated vertices;
}
```

The test takes $d(a)+d(b)$ time (so is of order n). The inside loop is executed no more than $|E|$ times. Since no more than $|E|$ edges can be removed, the outside loop can be executed no more than $|E|$ times. So the total work is bounded above by $n|E|^2 + n$, with the last contribution coming from the removal of isolates. An improved method, shown later, takes $\sum d^2(v)$ work.

⁴ A simple undirected graph has edges that lack direction, has no edges joining any vertex to itself, and has no more than one edge between any pair of vertices.

Observation: Each clique of order k in G is contained in a k -truss of G . This was by design, since trusses were defined by relaxing cliques.

This observation is useful if one really *does* want to enumerate cliques. The process can be accelerated by first finding the trusses, then looking for the cliques within the trusses.

Observation: A k -truss need not contain a clique of order k .

One might suppose that a k -truss is composed of one or more intersecting cliques of order k . This is not necessarily so. Figure 2 shows an example for $k = 4$, in which no clique of order 4 appears, even though each edge closes two triangles.

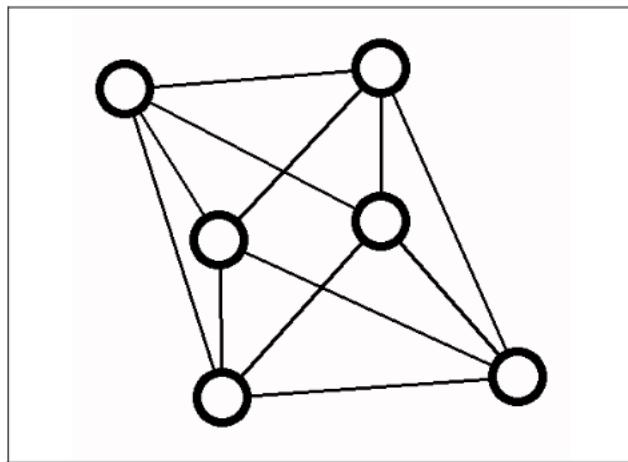


Figure 2. A k -Truss Need Not Contain a Clique of Order k . Here is an example for $k = 4$.

Prop: Every vertex v in a k -truss has $d(v) \geq k-1$.

PROOF

Proof: Choose vertex v in a k -truss T . Since trusses contain no isolates, v must have an edge (v, v^*) in T . By the definition of a truss, vertices v and v^* must share at least $k-2$ neighbors distinct from v and v^* . These, added to v^* , constitute at least $k-1$ vertices adjacent to v .

Observation: For $k > 2$, each k -truss of G is the subgraph of a $(k-1)$ -truss of G .

The result of this observation is useful for computation purposes. If one wanted to find all k -trusses for $k = 3, \dots, m$, one could first find the 3-trusses, then the 4-trusses from the 3-trusses, and so on.

Prop: Each k -truss of G is a subgraph of a $(k-1)$ -core of G .

PROOF

Let T be a k -truss of G . Each vertex in T is adjacent to at least $k-1$ vertices in T . So T qualifies as a $(k-1)$ -core of G , except, perhaps, for maximality. The maximality of cores requires that all of T be in the same core.

This proposition and the observation that each clique of order k is contained in a k -truss support the idea that the truss is someplace between the clique, which is too restrictive, and the core, which is too lax.

Prop: Each k -truss of G has an edge connectivity⁵ $\kappa_e \geq k - 1$.

**P
R
O
O
F**

Let C be a minimal cut set⁶ of edges in k -truss T . Let $e = (a, b) \in C$. When all of the edges in C are removed, the vertices a and b will be in separate components, else the removal of e was unnecessary, violating the minimality of C . Since a and b are also joined by $k-2$ independent paths of length 2, a minimum of $k-2$ edges must also be removed to place a and b in separate components. Thus, $|C| \geq k - 1$.

Observation: The converse of above proposition is not true, that is, a graph with edge connectivity $\kappa_e = k - 1$ is not necessarily a k -truss.

A 4-cycle, for example, has $\kappa_e = 2$, but is not a 3-truss.

Observation: The vertex connectivity⁷ of a k -truss can be as low as 1, no matter how high k is.

As an example, a larger k -truss can be created by joining two k -trusses by a common member, such is pictured in Figure 3.

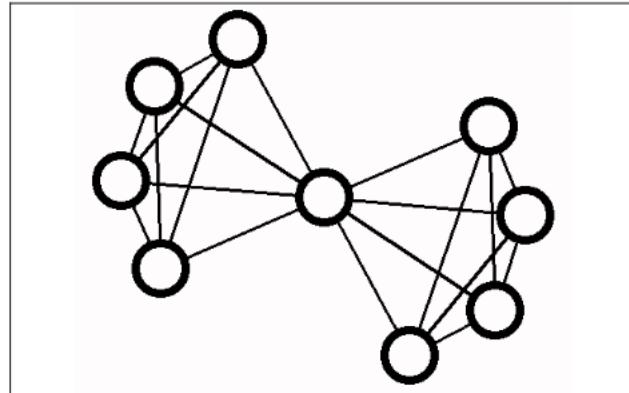


Figure 3. A k -Truss Can Have Vertex Connectivity of 1.

⁵ The edge connectivity of a graph is the minimum number of edges whose removal will divide the graph into separate components.

⁶ A cut set is a set of edges whose removal will divide the graph into separate components.

⁷ The vertex connectivity of a graph is the minimum number of vertices whose removal will divide the graph into separate components.

Prop: A cutpoint⁸ of a k -truss joins k -trusses.

P
R
O
O
F

Let v be a cutpoint of k -truss T , and let it join the subgraphs C_1, \dots, C_b , each of which includes v . Each edge in C_1 joins two vertices of T that share at least $k-2$ common neighbors, each of which must be in C_1 or violate the selection of v as a cutpoint. So C_1 is a k -truss. Similarly, C_2, \dots, C_b are k -trusses.

Corollary: If a k -truss contains a cutpoint v , it has $d(v) \geq 2(k - 1)$.

P
F

By the proposition, the cutpoint joins at least 2 edge-disjoint k -trusses, each of which must contribute $k-1$ to the degree.

⁸ A cutpoint is a vertex whose removal will break the graph into components.

Prop: Suppose that a k -truss contains m vertices and has diameter d . Then

$$m \geq \begin{cases} \frac{d+1}{2}k, & d \text{ odd} \\ \frac{d}{2}k + 1, & d \text{ even} \end{cases}.$$

**P
R
O
O
F**

Let the path (v_0, v_1, \dots, v_d) be a diameter⁹ of the truss. This is illustrated in Figure 3. Each of the edges in the diameter must be supported by at least $k-2$ paths of length 2, that is, each pair (v_i, v_{i+1}) must have at least $k-2$ common neighbors. The required common neighbors are contained in the sets $\{B_i\}$ and $\{T_i\}$ in the figure: each B_i holds those vertices not in the diameter that are adjacent to both v_i and v_{i+1} , and to no other vertices in the diameter; each T_i holds those vertices that are adjacent to all three of v_{i-1} , v_i and v_{i+1} . Note that the $\{T_i\}$ sets are disjoint, else the shown path would be longer than the diameter. A vertex placed in one of the $\{T_i\}$ serves double duty by reinforcing two of the diameter edges. For purposes of finding the minimum of m , assume that the sets contain no unnecessary members.

The size of the truss must include the members of the sets in Figure 3 and the diameter, that is, $m \geq d + 1 + \sum_{i=0}^{d-1} |B_i| + \sum_{i=1}^{d-1} |T_i|$. For each i , $|B_i|$ must be $k-2$, diminished by the number of vertices in the opposing $\{T_i\}$, so that $\sum_{i=0}^{d-1} |B_i| \geq d(k-2) - 2 \sum_{i=1}^{d-1} |T_i|$ and $m \geq 1 + d(k-1) - \sum_{i=1}^{d-1} |T_i|$. The set sizes are constrained by $0 \leq |T_i| \leq k-2$, $i = 1, \dots, d-1$ and by $0 \leq |T_i| + |T_{i+1}| \leq k-2$, $i = 1, \dots, d-2$, so that $0 \leq \sum_{i=1}^{d-1} |T_i| \leq \text{floor}(d/2)(k-2)$. Finally, $m \geq 1 + d(k-1) - \text{floor}(d/2)(k-2)$, or

$$m \geq \begin{cases} \frac{dk}{2} + 1, & n \text{ even} \\ \frac{dk}{2} + \frac{k}{2}, & n \text{ odd} \end{cases}.$$

⁹ Diameter, when a path as opposed to an integer, is a shortest path going between two vertices that are at a distance of the graph diameter from each other.

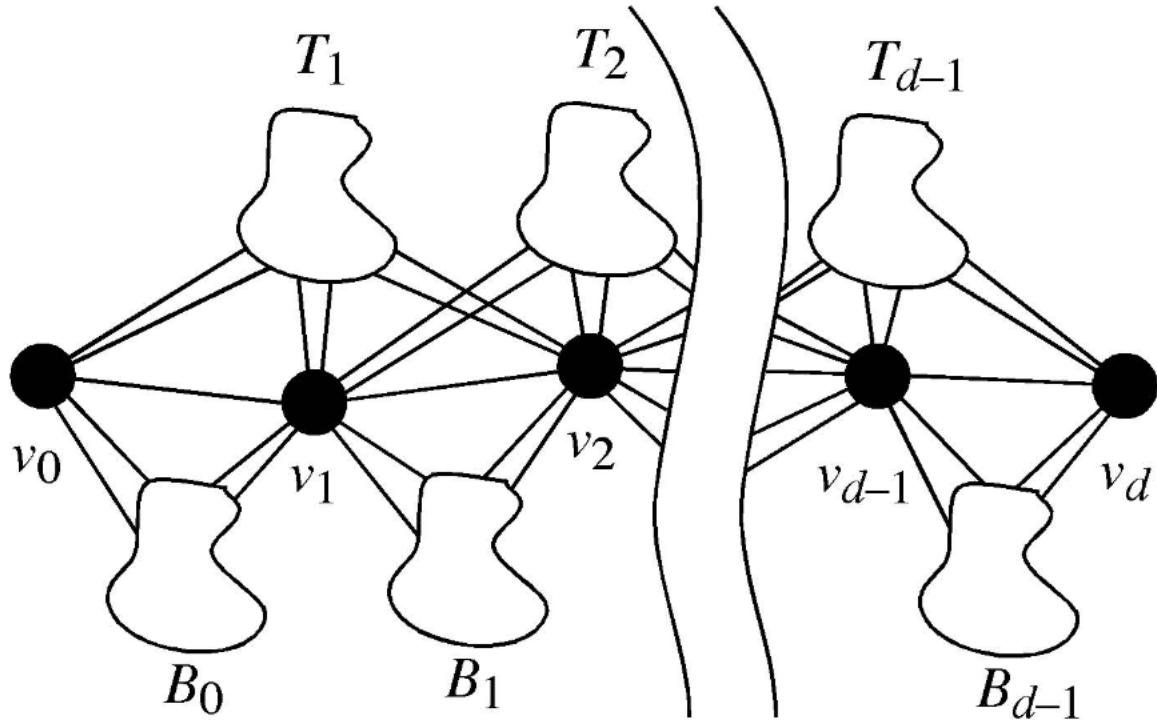


Figure 3. A Diameter of the Truss. Each of the edges in the diameter must be paralleled by $k-2$ paths of length 2 through intermediaries held in the sets shown to either side of the diameter.

Observation: The inequality of above is sharp.

CONSTRUCTION

Demonstration by construction:

For even diameter, construct a graph such as in Figure 4, in which cliques of k elements share complete subgraphs of $k-1$ elements. The number of vertices reaches the minimum $dk/2+1$.

For odd diameter, construct a graph such as in Figure 5, in which cliques of k elements share complete subgraphs of $k-1$ elements, with an additional order- k clique at the end. The number of vertices reaches the minimum $(d+1)k/2$.

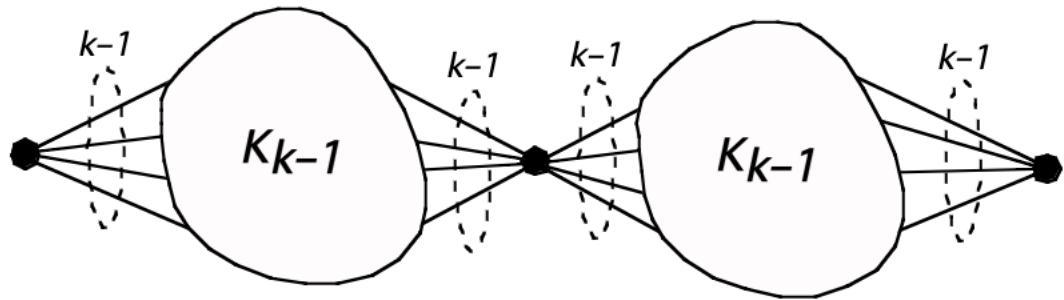


Figure 4. A Diameter-4 k -Truss with $dk/2+1$ vertices. The diameter passes through 4 cliques of k members, each sharing a complete subgraph of $k-1$ members with another.

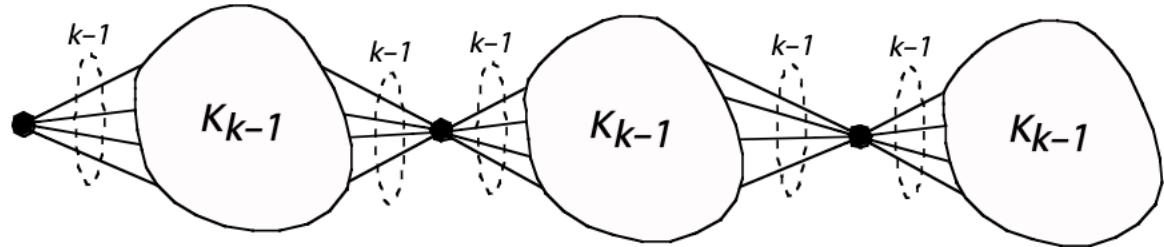


Figure 5. A Diameter-5 k -Truss with $(d+1)k/2$ vertices. The diameter passes through 4 cliques of k members, each sharing a complete subgraph of $k-1$ members with another, and ending with an additional clique of order k .

Finding Maximal Trusses

Given a simple graph G , the maximal k -truss of G can be found simply by removing from G those elements that are not part of the maximal k -truss.

A summary of the approach is shown in Snippet 1. First, the graph is reduced to its maximal $k-1$ core. This is a cheap way of dispatching many vertices and their adjacent edges before doing the harder work of eliminating edges of insufficient support. Most of the work is described in Snippets 2 and 3. Once inappropriate edges are removed, isolated nodes are likely to result; they are removed as a last step.

```
reduceGraphToKTruss( G, k )
{
    //      remove obvious vertices by taking to k-1 core
    reduceToKCore( G, k-1 );

    //      remove edges not supported by k-2 edge pairs
    removeUnsupportedEdges( G, k-2 );

    //      iteratively remove vertices of insufficient valence
    removeIsolatedVertices( G );
}
```

Snippet 1. Part One of a Sketch of Code to Find k -Trusses. The bulk of the code is shown in Snippets 2 and 3. Removing isolates is obvious and not amplified further.

Snippet 2 shows a sketch of reducing G to its maximal j -core. Vertices are removed if their valence is less than j . As vertices are removed, their neighbors lose valence and must be tested as well. Removal is through a queue of vertices (a set would do). The set is managed such that an attempt to add a duplicate is detected in order(1) time and avoided. Work of order($n + |E|$) is done in the initialization. Note that the inner loop in the removal process can be executed no more than $\sum_{v \in G} d(v) = 2|E|$ times, so that the overall work for Snippet 2 is of order($n + |E|$).

```

reduceToKCore( G, j )
{
    //      init degree counter
     $\forall v \in G, D(v) \leftarrow 0;$ 
     $\forall (a,b) \in G, \text{ increment } D(a) \text{ and } D(b);$ 

    //      init vertex queue
     $q \leftarrow \emptyset;$ 
     $\forall v \in G, \text{ if } D(v) < j \text{ then } q \leftarrow q \cup \{v\};$ 

    //      iteratively remove vertices of insufficient valence
    while  $q \neq \emptyset$  do
        {
            pull  $v$  from  $q$ ;
             $\forall a \in N(v),$ 
            {
                decrement  $D(a)$ ;
                if  $D(a) < j, q \leftarrow q \cup \{a\};$ 
            };
            remove  $v$  from  $G$ ;
        };
}

```

Snippet 2. Part Two of a Sketch of Code to Find k -Trusses. This portion reduces G to its maximal j -core. The queue is smart enough not to duplicate members and to reject duplicates in order(1) time.

The heart of the process¹⁰ is outlined in Snippet 3, which removes from G any edge that is not supported by at least j edge pairs, that is, any edge that does not complete at least j triangles. Initially, all edges are examined; as edges are removed, neighboring edges need to be reconsidered. Those edges to be examined are held in a set that is managed in such a way that attempted duplicate entries are rejected in order(1) time.

A representative edge (a,b) is determined to be sufficiently supported if vertices a and b have at least j common neighbors. That test takes $d(a) + d(b)$ work. If the test fails, the amount of work to queue neighboring edges and remove the edge from G is also $d(a) + d(b)$. Edge (a,b) can be queued at most one time. So the work goes as

$$\sum_{e=(a,b)} [d(a) + d(b)] = 2 \sum_{v \in G} d^2(v), \text{ that is, the work is of order } \sum_{v \in G} d^2(v).$$

¹⁰ The approach in Snippet 3 was suggested by Michael Ferguson, and improves on the author's original algorithm.

```

removeUnsupportedEdges(  $G, j$  )
{
    //      this will hold edges to remove
     $q \leftarrow \emptyset;$ 

    //      count triangles  $C$  supporting each edge
    //      if count too small, queue edge for removal
     $\forall e = (a, b) \in E$  do
    {
        put members of  $N(a)$  in a hash table  $T$ ;
         $c \leftarrow 0$ ;
         $\forall v \in N(b)$ , if  $v \in T$  then  $c \leftarrow c + 1$ ;
        if  $c < j$ , then  $\{q \leftarrow q \cup e\};$  remove  $e$  from  $G$ ; }
        else  $C(e) \leftarrow c$ ;
    };

    //      remove queued edges, perhaps queuing neighboring ones as well
    while  $q \neq \emptyset$  do
    {
        pull  $e = (a, b)$  from  $q$ ;
        put members of  $N(a)$  in a hash table  $T$ ;
         $I \leftarrow \emptyset$ ;
         $\forall v \in N(b)$ , if  $v \in T$  then  $I \leftarrow I \cup \{v\}$ ;
         $\forall e'$  joining  $a$  or  $b$  to an element of  $I$ , do
        {
            decrement  $C(e')$ ;
            if  $C(e') < j$ , then  $\{q \leftarrow q \cup e'\};$  remove  $e'$  from  $G$ ; }
        };
    };
}

```

Snippet 3. Part Three of a Sketch of Code to Find k -Trusses. This portion does most of the work, and removes from G those edges that do not complete at least j triangles. The queue (set) is smart enough not to duplicate members and to reject duplicates in order(1) time.

Note that Snippet 3 is really carried out on the $k-1$ core of G , rather than the original G . For practical cases, the work of finding k -trusses will be dominated by the code of Snippet 3, but much improved by the code of Snippet 2.

As noted earlier, if one wants to compute k -trusses for a range of k values, one can take a $k-1$ -truss as an input to the code outlined in this session to obtain a k -truss, with less work than starting from scratch.

Examples

Krackhardt's high-tech managers

Krackhardt (1987) gathered information from managers of a high-tech manufacturing firm in the west coast of the U.S. The graphs here were based on the “seeks advice from” relationship among the managers, taken from Wasserman and Faust (1994). Figure 6 shows the reciprocated advice relationship, suggesting trust. In Figure 6(a), the entire graph is shown. Figure 6(b) shows the 4-truss of (a), with the remainder dimmed. Figure 6(c) is the corresponding 3-core. Finally, Figure 6(d) lists the cliques containing more than 3 actors, all of which are of order 4. The truss does a nice job of showing the cohesive subgraph that captures all of the cliques in a clear representation that lacks the added clutter of the 3-core.

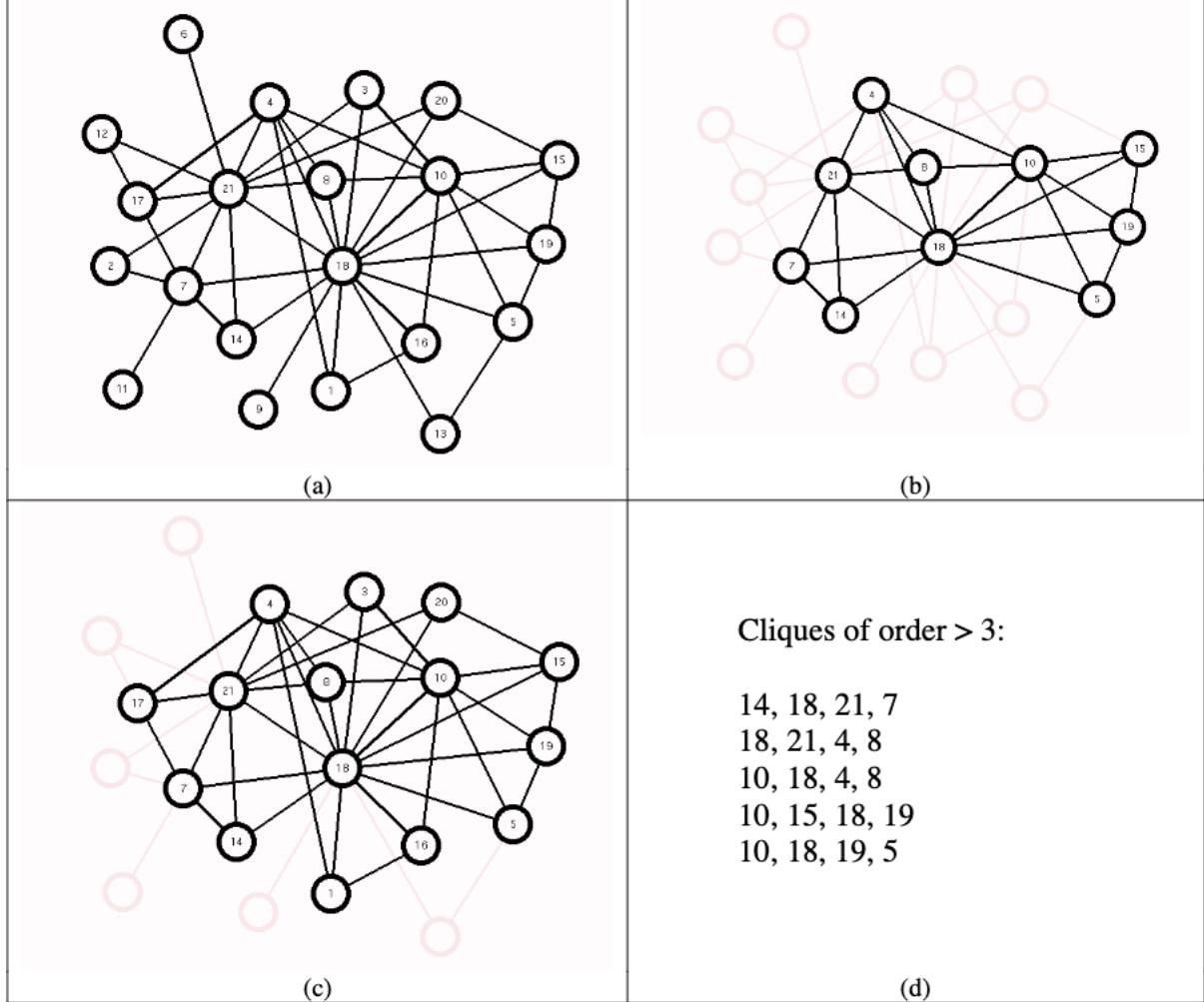


Figure 6. Krackhardt's High-Tech Managers (a) Reciprocated Advice, (b) 4-truss of (a), (c) 3-core of (a). (d) cliques of order 4 and larger. No 5-trusses exist.

Freeman's EIES researchers

This data was gathered by S. Freeman and L. Freeman (and reported in part by Wasserman and Faust (1994), from which the author obtained his data) at a computer conference of social network research. Included were personal relationship information among the participating researchers and the number of times each researcher communicated with another through a then-novel electronic mailing system.

Figures 7 and 8 are based on personal familiarity of the actors as reported at the end of the study. A link in the diagrams indicates that the joined actors each labeled the other as a friend or as someone with whom they had met. Figure 7(a) shows the entire network. Figures 7(b), 7(c), and 7(d) show the maximal 3-truss, 4-trusses, and 5-truss, respectively. Figure 7(c) represents 10 cliques of order 4 in a compact, uncluttered form.

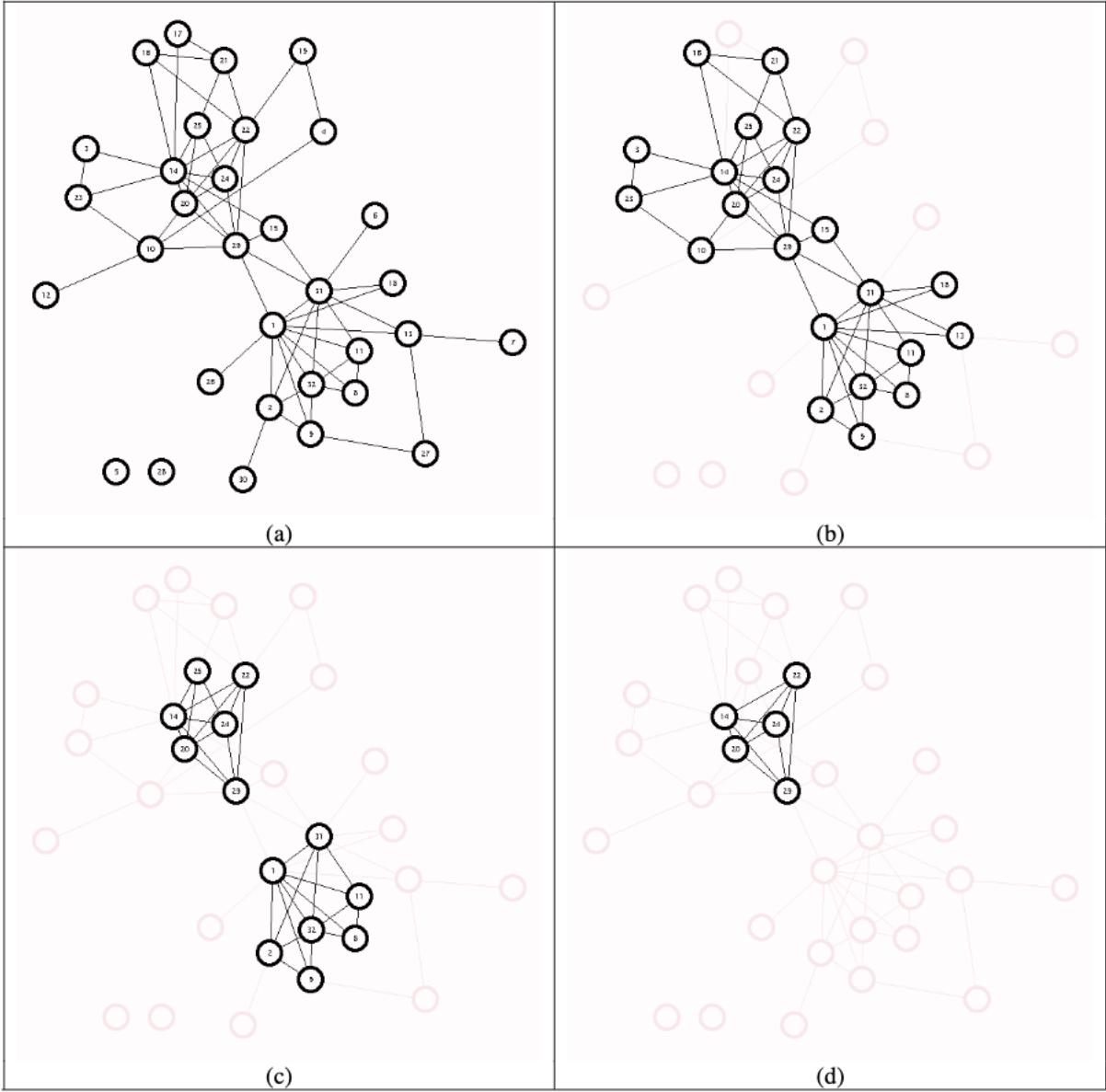


Figure 7. Trusses for Reciprocated Friendship and Meeting Relations for Freeman's EIES Researchers at Time 2. (a) total subgraph, (b) maximal 3-truss containing 35 cliques of order 3, (c) maximal 4-trusses containing 10 cliques of order 4, (d) maximal 5-truss consisting of one clique of order 5. No 6-trusses exist. The cliques counted here were not necessarily maximal.

Figure 8 is based on the same data as Figure 7, but shows k -cores. The 2-core is a bit bigger than the 3-truss (as it should be), and the 4-core is the same as the 5-truss. The difference between the 3-core and the 4-truss is significant, in that the much weaker connections retained in the 3-core are suppressed in the 4-truss.

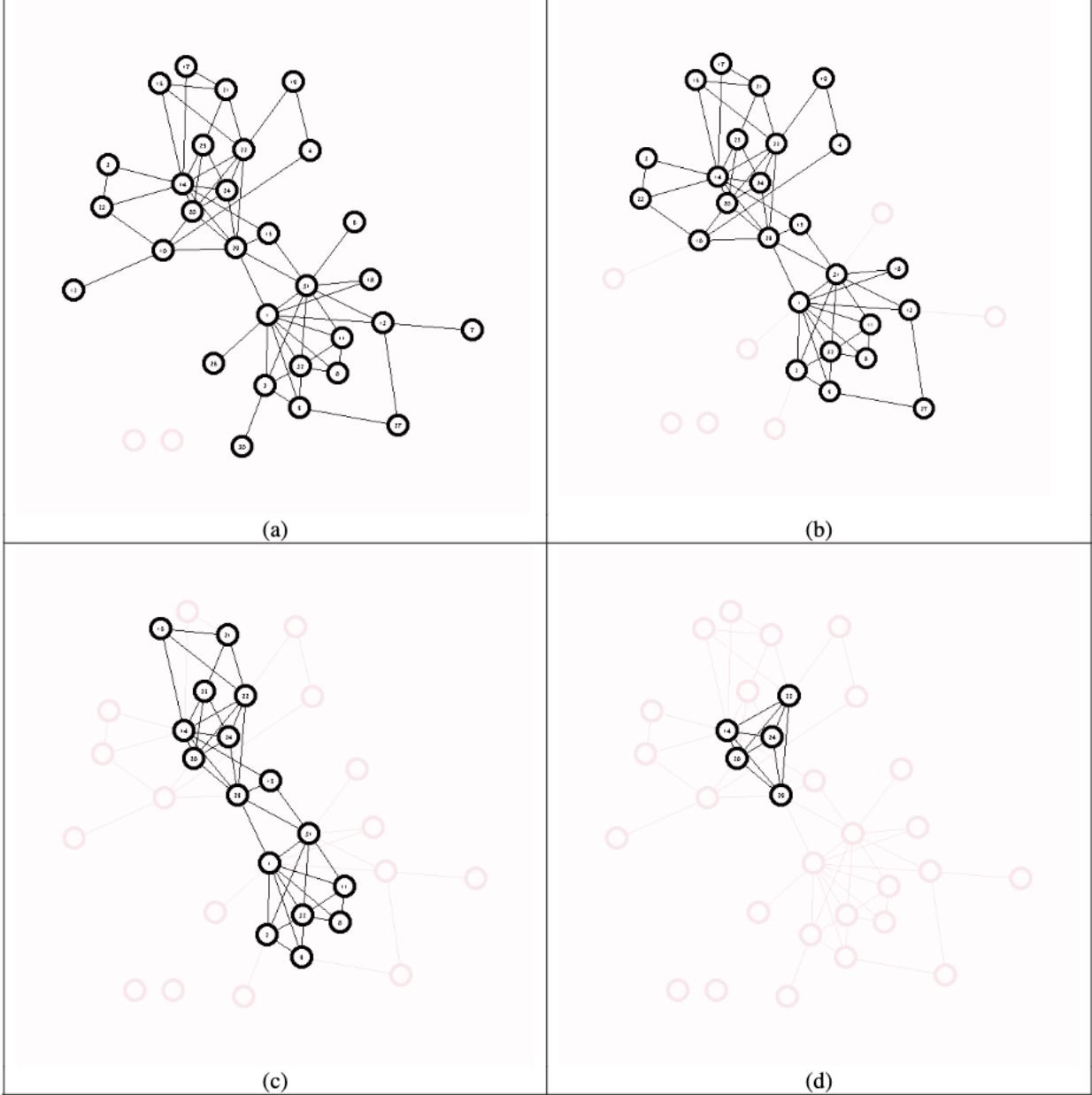


Figure 8. Cores for Reciprocated Friendship and Meeting Relations for Freeman's EIES Researchers at Time 2. (a) 1-core, (b) 2-core, (c) 3-core, (d) 4-core. No 5-cores exist.

Figure 9 is built from electronic communications between the actors. The graph of Figure 9(a) shows a link between actors A and B if A sent more than 4 messages to B and B sent more than 4 messages to A . Figure 9(b) shows the 10-truss of 9(a), which has been rearranged and pictured by itself in 9(c). As Figure 9(c) suggests, the 10-truss looks much like a clique. In fact, the truss consists of 63 edges — only 3 shy of a clique of its 12 members. This truss is a highly-overlapped set of three cliques of order 10 (listed in Figure 9(d)). For analytical purposes, the single truss is more informative than an enumeration of the cliques.

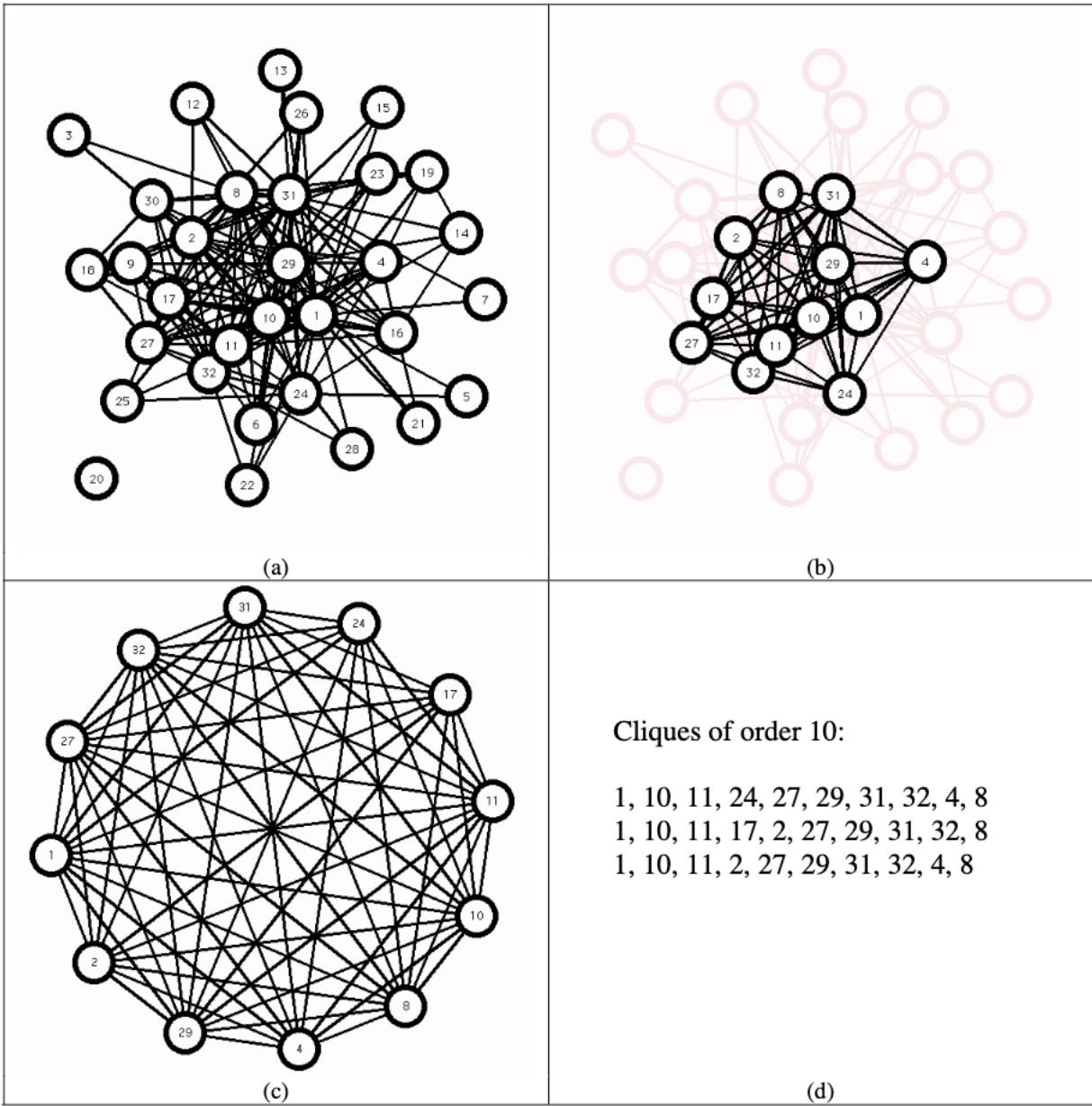


Figure 9. Reciprocated Electronic Communication for Freeman's EIES Researchers. (a) all reciprocated contacts of more than 4 times, (b) maximal 10-truss of reciprocated contacts of more than 4 times, (c) rearrangement of the truss of (b), (d) cliques of order 10 in (a). No 11-trusses are present.

Magazine article similarity graph

This example considers relationships among 135 text documents extracted from English-language magazines circa 1990. The relationships are represented by a graph in

which the documents are vertices, and edges join documents judged to be similar. Similarity was based on overlapping 5-gram spectra of the documents (see Damashek 1995). The graph is a single component. Figures 10, 11, and 12 show trusses from this graph. One would hope that trusses would capture topical groupings. In the figures, vertices are labeled with automatically-generated highlights (Cohen 1995) to suggest their topic.

Figure 10 shows the four maximal 5 trusses in the magazines graph. The one in the upper left is evidently about lead poisoning in children (and is a clique of order 6), the one in the upper right combines documents about the confirmation of judge Clarence Thomas and race relations in Africa, the truss in the lower left combines USSR troubles and the breakup of Yugoslavia (an intermingling of five cliques of order 5), and the one on the bottom right discusses school choice.

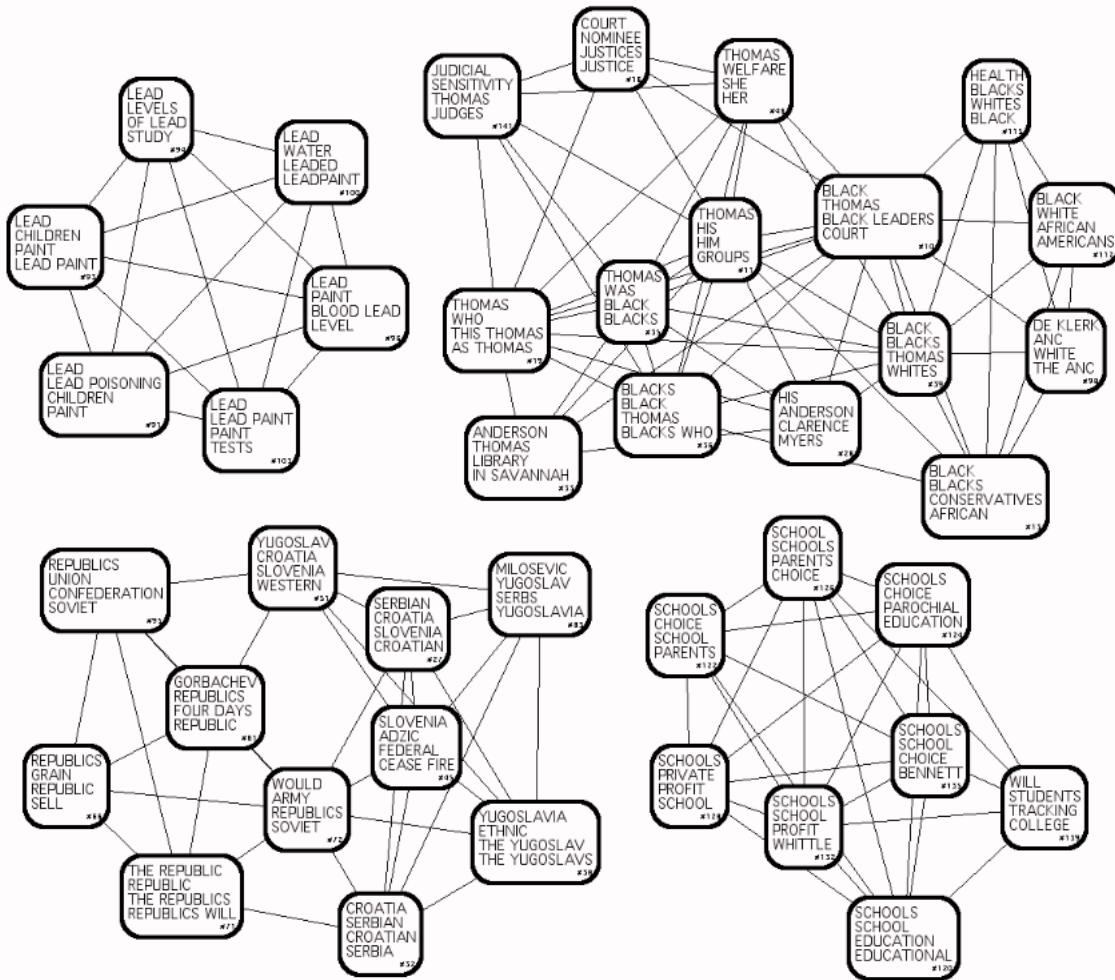


Figure 10. Maximal 5-Trusses From the Magazines Graph. Each vertex represents a text document drawn from a magazine article and labeled with some highlights. Edges join vertices whose documents were judged to be similar.

Figure 11 shows the maximal 6-trusses for the same graph, completely eliminating one of the 5-trusses and refining another. The 7-trusses (both cliques) are shown in Figure 12.

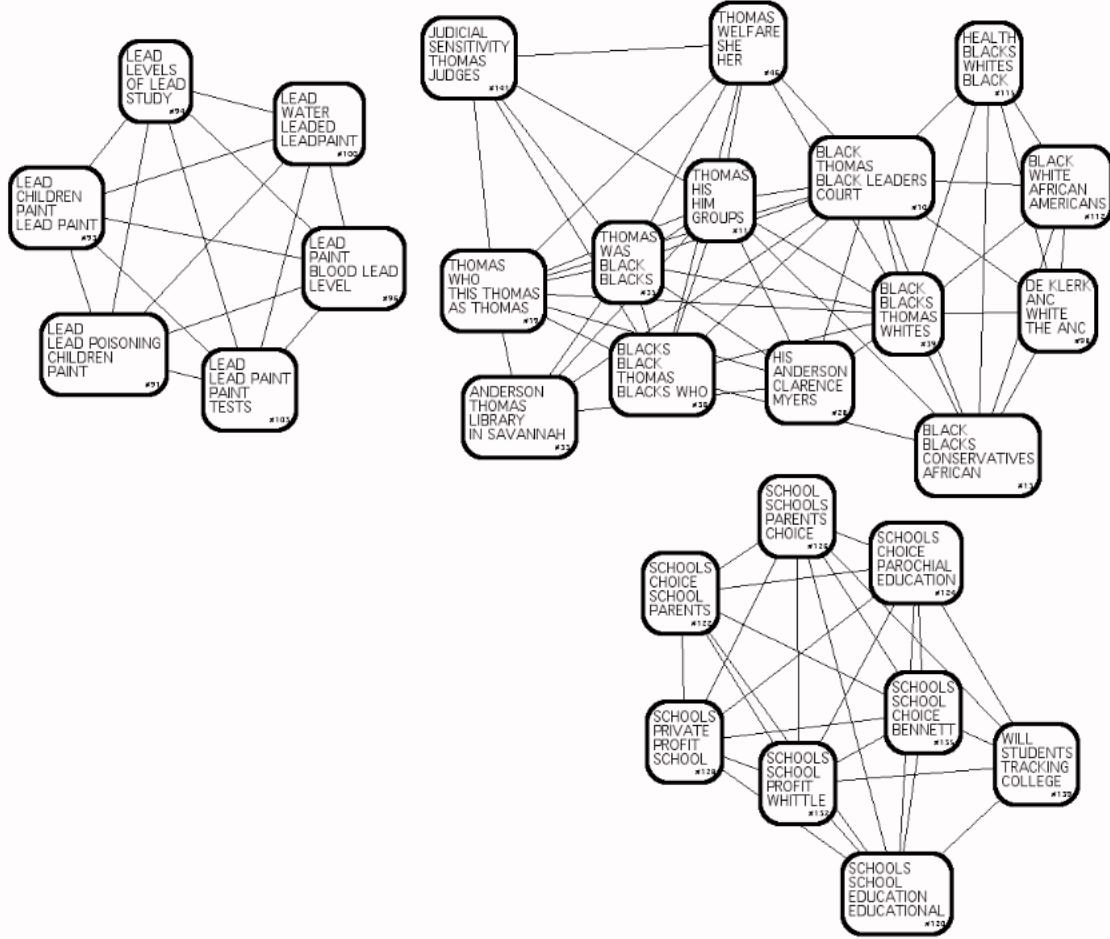


Figure 11. Maximal 6-Trusses From the Magazines Graph. Each vertex represents a text document drawn from a magazine article and labeled with some highlights. Edges join vertices whose documents were judged to be similar.

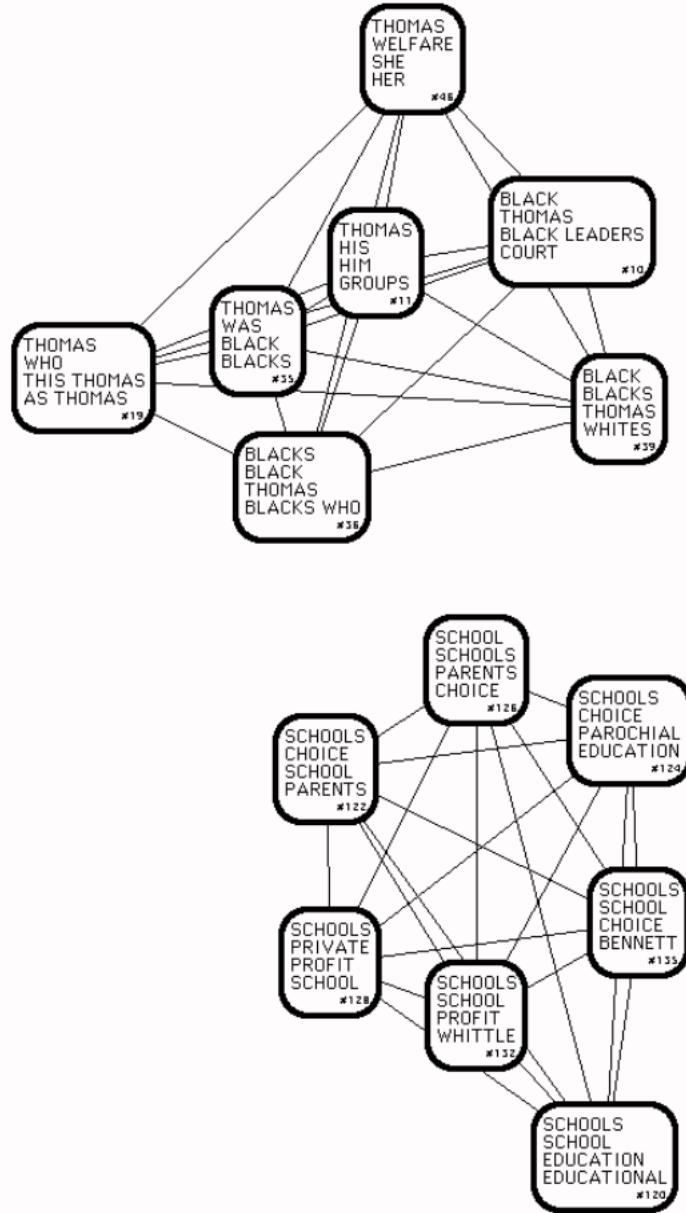


Figure 12. Maximal 7-Trusses From the Magazines Graph. Both are cliques of order 7.

On this data, trusses seem to do a good job of identifying topical groupings, though the size of the trusses to seek is a question.

Concluding Remarks

The k -truss provides a nice compromise between the too-promiscuous $(k-1)$ -core and the too-strict clique of order k . Unlike the clique and other relaxations of it, the k -truss is efficiently computed. Like the $(k-1)$ -core, the k -truss reports interlocking groups as a single subgraph, rather than requiring distracting enumeration. Finally, the truss is built upon the intuitively satisfying observation that strong real-life ties are likely to be supported by other ties through single intermediaries.

Acknowledgements

The author would like to thank Kimberly Glasgow and Steve Poulos for reviewing the manuscript. He is also indebted to Michael Ferguson for pointing out a better way of constructing trusses.

References

- Alba (1973): Alba, R.D., “A graph-theoretic definition of a sociometric clique,” *Journal of Mathematical Sociology*, Vol. 3, 1973, pp. 113-126.
- Borgatti, Everett, and Shirey (1990): Borgatti, S.P., Everett, M.G., and Shirey, P.R., “LS sets, lambda sets and other cohesive subsets,” *Social Networks*, Vol., 12, 1990, pp. 337-357.
- Bron and Kerbosch 1973: Bron, C., and Kerbosch, J., “Finding all cliques of an undirected graph,” *Communications of the ACM*, Vol. 16, No. 9, September 1973, pp. 575-577.
- Cohen (1995): Cohen, J. “Highlights: language- and domain-independent automatic indexing terms for abstracting,” *Journal of the American Society for Information Science*, Vol. 46, No. 3, pp. 162-174, and erratum, Vol. 47, No. 3, pg. 260.
- Damashek (1995): Damashek, M., “Gauging similarity with n-grams: language-independent categorization of text,” *Science*, Vol. 267, 10 February, 1995, pp. 843-848.

- Girvan and Newman (2001): Girvan, M. and Newman, M. E. J., "Community structure in social and biological networks," *Proc. National Acad. Sci. USA*, 99, 2003, pp. 7821-7826.
- Krackhardt (1987): Krackhardt, D., "Cognitive social structures," *Social Networks*, Vol. 9, pp. 109-134.
- Luce and Perry (1949): Luce, R.D., and Perry, A.D., "A method of matrix analysis of group structure," *Psychometrika*, Vol. 14, No. 1, March 1949, pp. 95-116.
- Luce (1950): Luce, R.D., "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, Vol. 15, No. 2, June 1950, pp. 169-190.
- Luccio and Sami (1969): Luccio, F., and Sami, M., "On the decomposition of networks in minimally interconnected subnetworks," *IEEE Transactions on Circuit Theory*, CT-16, 1969, pp. 184-188.
- Mokken (1979): Mokken, R.J., "Cliques, clubs, and clans," *Quality and Quantity*, Vol. 13, 1979, pp. 161-173.
- Moody (2001): Moody, J., "Peer influence groups: identifying dense clusters in large networks," *Social Networks*, Vol. 23, 2001, pp. 261-283.
- Moody and White (2003): Moody, J., and White, D. R., "Structural cohesion and embeddedness: a hierarchical concept of social groups," *American Sociological Review*, Vol. 68, Feb 2003, pp. 103-127.
- Scott (1991): Scott, J., *Social Network Analysis: A Handbook*, 2nd Ed., SAGE Publications, Ltd., London, 2000.
- Seidman (1983): Seidman, S.B., "Network structure and minimum degree," *Social Networks*, Vol. 5, 1983, pp. 269-287.
- Seidman and Foster (1978): Seidman, S.B., and Foster, B.L., "A graph-theoretic generalization of the clique concept," *Journal of Mathematical Sociology*, Vol. 6, 1978, pp. 139-154.
- Wasserman and Faust (1994): Wasserman, S. and Faust., K, *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.

White and Harary (2001):

White, D. R., and Harary, F., "The cohesiveness of blocks in social networks: node connectivity and conditional density," *Sociological Methodology*, Vol. 31, pp. 305-359, 2001.