Social Networks 10 (1988) 77–91
North-Holland

CALCULATING ROLE SIMILARITIES: AN ALGORITHM THAT HELPS DETERMINE THE ORBITS OF A GRAPH

Martin G. EVERETT *
Thames Polytechnic

Steve BORGATTI **
University of California, Irvine

The orbits or a graph, digraph or network provide an effective definition for role equivalence since they are a natural generalization of the principle of substitutability of structural equivalence. Calculation of the orbits is a computationally difficult task but in this paper we present a fast and efficient algorithm which finds the orbits of a large class of graphs. In addition, we suggest a simple measure of role similarity based upon the constructions contained within the algorithm. This makes it possible to perform a role analysis when only a limited number of automorphisms exist.

1. Introduction

Modelling techniques have recently been developed which try to capture the notion of role equivalence. The first attempt was by Lorrain and White (1971), who defined structural equivalence between pairs of actors. Certain fundamental difficulties with this definition were pointed out by Sailer (1978) and White and Reitz (1983). In essence, their criticism was based on the fact that actors playing the same role do so by exhibiting similar relational patterns to *different* actors playing equivalent roles. In Lorrain and White's definition, actors playing the same role have the same relationship to the *same* actors.

The critics propose alternative definitions which unfortunately lose the important principle of substitutability. Structural equivalence as presented by Lorrain and White is based upon the fact that two

* School of Mathematics, Statistics and Computing, Thames Polytechnic, Wellington Street, London SE18 6PF, U.K.
** School of Social Sciences, University of California, Irvine, CA 92627, U.S.A.

structurally equivalent actors can be substituted for each other without changing the structure of the social network. A different way of viewing this concept is as follows: consider a social network in which each actor except two is labelled. If there is no way to distinguish between these two nodes then they are structurally equivalent. Everett (1985) proposes a natural generalization of this concept which simultaneously retains the basic principle of substitutability and answers the criticism of Sailer, and White and Reitz. If we examine the alternate way of viewing structural equivalence described above then it is the labelling process which restricts structural equivalence. By labelling all but two vertices we immediately have a technique for identifying otherwise indistinguishable vertices. The labelled vertices mean that structural equivalence can only be applied to vertices which connect to the same vertices in the same way. If we remove *all* labels and group together indistinguishable vertices then we obtain a natural generalization of structural equivalence in which the principle of substitution is retained and which simultaneously answers the criticisms of Sailer (1978) and White and Reitz (1983).

To illustrate the above, consider the simple network shown in Figure 1. The vertices 1 and 2 are structurally equivalent since if we remove the labels from these nodes then we would not know which was which. Alternatively, if we interchange the labels of 1 and 2 the new graph would be isomorphic to the graph of Figure 1. The same is true of the vertices 3 and 4; however, we note that 2 and 3 are not structurally equivalent since if we remove these two labels then we can easily replace them by noting that 2 is adjacent to 5, and 3 is adjacent to 6. This time if we interchange the labels 2 and 3, the resultant graph is not isomorphic to the graph in Figure 1. Similarly, vertices 5 and 6 are not structurally equivalent.

If we start from an unlabelled version of the graph in Figure 1 then there is no way to distinguish between the vertices within the sets

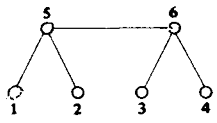

Fig. 1.

{1, 2, 3, 4} and {5, 6}. For example, if we wish to assign label 5 to the unlabelled graph, then we would have two possibilities, namely either of the two degree-3 vertices. There is, however, no way of knowing which of these is 5 without additional information - such as the labelling of another vertex. In Lorrain and White's version, this information is provided and so 5 and 6 fail to be structurally equivalent.

The sets {1, 2, 3, 4} and {5, 6} are known as the orbits of the graph and Everett (1985) argues that these are the sets of role-equivalence actors. Winship and Mandel (1983) also use orbits as the mathematical basis of role similarity but they quickly discard the concept in favor of their own role equivalence. They do refer to further work on orbits but unfortunately this work has never been published. Following the terminology of Winship and Mandel we shall say that two vertices in the same orbit are automorphically equivalent.

It should be noted that in the above example the degrees of the vertices were sufficient to find the orbits. In general any property of the graph may be used to distinguish vertices or sets of vertices in an attempt to establish the orbits. Since there are no known complete sets of graphical invariants, it is not possible to find the orbits by examining a particular set of properties. Consequently the only algorithm which can find orbits must be based on a direct search technique and since this search must be based on permutations it will soon become computationally infeasible even for small graphs. Everett (1985) suggests that finding the orbits is a relatively easy task once the adjacency matrix of the graph has been reduced to Jordan canonical form. Unfortunately, this is not the case and the reduction technique merely complicates rather than simplifies any direct search technique. In this paper we shall describe an algorithm which finds orbits efficiently for a large class of graphs. In addition, we extend the basic algorithm to give a simple measure of the degree of role equivalence.

2. Mathematical definitions

For the sake of simplicity we shall initially consider only graphs, leaving the extension to digraphs and networks for a later section.

Let $G(V, E)$ be a graph with vertex set V and edge set E. An automorphism of G is a permutation π of V with the property that $(a, b) \in E$ if and only if $(\pi(a), \pi(b)) \in E$. Two vertices $a, b \in V$

belong to the same orbit of G if and only if there exists an automorphism π such that $\pi(a) = b$. Belonging to the same orbit is an equivalence relation and hence the vertices of G are partitioned into distinct equivalence classes.

If S is a subset of V then the *neighborhood* $N(S)$ is defined by

$$N(S) = S \cup \{ v \in V - S \colon (v, s) \in E,\ s \in S \}.$$

We define the higher-order neighborhoods inductively

$$N^{i+1}(S) = N(N^i(S)).$$

Consider the graph in Figure 1. If $S = \{1\}$ then

$$N(S) = \{1, 5\}$$

$$N^2(S) = \{1, 2, 5, 6\}$$

$$N^3(S) = \{1, 2, 3, 4, 5, 6\}.$$

If $x \in V$ then we define the *point-deleted neighborhood* $\tilde{N}^i(x)$ as $N^i(x) - \{x\}$. The *degree vector* of a graph G is defined by

$$d(G) = (d(v_1), d(v_2), \ldots, d(v_n)) \quad \text{for } v_i \in V(G)$$

where $d(v_i)$ is the degree of the vertex v_i,

$$n = |V|,$$

$$d(v_i) \le d(v_{i+1}), \quad \text{for } i = 1, \ldots, n-1.$$

If S is a subset of V then $\langle S \rangle$ is the induced subgraph, *i.e.* the maximal subgraph of G with vertex set S. We shall write $d(S)$ for $d(\langle S \rangle)$. Hence, in our example,

$$d(\tilde{N}(S)) = (0)$$

$$d(\tilde{N}^2(S)) = (1, 1, 2).$$

The problem of deciding whether two graphs are isomorphic has been the subject of much research. So far no polynomial time algorithm has been discovered for solving this problem. In fact, it is not even known whether this problem is NP-complete or not (though it is clearly NP) and hence the subject has received particular attention in recent years. Fontet (1975) proved that finding the orbits solves the graph isomorphism problem. It therefore seems highly unlikely that a polynomial-time algorithm exists for this problem. For certain classes of graphs polynomial time algorithms for finding orbits do exist, for example trees (Fontet 1976), interval graphs, planar graphs, and outer-planar graphs (Colbourn and Booth 1981). (In fact, linear-time algorithms exist for all these classes.)

3. The algorithm

The orbits of a graph are a set of vertices that are indistinguishable except for labelling. Hence if any two vertices can be differentiated by any property then they must be in different orbits. Unfortunately, there is not a set of properties which completely describes a graph. Consequently it is not possible to find the orbits by investigating a collection of attributes of each vertex. However, any property not shared by two vertices guarantees that they are in separate orbits. The algorithm presented here is based on this fact so that the actual orbits can only be a finer partition than that produced by the algorithm.

To illustrate the technique we start with a simple example. Consider the graph in Figure 1. Partition the vertex set so that we group together vertices of the same degree. We obtain the sets {1, 2, 3, 4} and {5, 6}. As it happens, these are the orbits, so that the degrees of the vertices were sufficient to characterize the orbits in this case. Unfortunately, it will not usually be that simple. Now let us examine the graph P_5 as shown in Figure 2.

The orbits are {1, 5}, {2, 4} and {3}. If we split the vertex set with respect to degree only we obtain the split {1, 5} and {2, 3, 4}. However

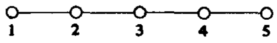

Fig. 2.

when we examine $d(\tilde{N}(x))$ for all $x \in \{2, 3, 4\}$ we obtain the following:

x	$\tilde{N}(x)$	$\tilde{N}^2(x)$	$d(\tilde{N}^2(x))$
2	{1, 3}	{1, 3, 4}	(0, 1, 1)
3	{2, 4}	{1, 2, 4, 5}	(1, 1, 1, 1)
4	{3, 5}	{2, 3, 5}	(0, 1, 1)

Since $d(\tilde{N}^2(2)) = d(\tilde{N}^2(4))$ but $\neq d(\tilde{N}^2(3))$ we can distinguish 3 from 2 and 4. Quite simply we are noting that 2 and 4 are directly adjacent to vertices of degrees 1 and 2 whereas 3 is adjacent to two vertices of degree 2.

In any graph we can continue to look at higher-order neighborhoods, so that x and y belong to the same set only if

$$d(\tilde{N}^i(x)) = d(\tilde{N}^i(y)) \quad \text{for } i \geq 0.$$

Note that the above technique combines structure with degree; the length and content of the degree vectors contain information not only about the degrees of the vertices but also about the basic construction of the graph. Consequently, this method can work even if the graph is regular.

Consider the cubic graph (every vertex has degree 3) shown in Figure 3. The construction of $d(\tilde{N}^2(x))$ for each vertex is as follows:

x	$\tilde{N}(x)$	$d(\tilde{N}(x))$
1	2, 3, 4	(1, 1, 2)
2	1, 3, 5	(0, 1, 1)
3	1, 2, 4	(1, 1, 2)
4	1, 3, 5	(0, 1, 1)
5	2, 4, 6	(0, 0, 0)
6	5, 7, 9	(0, 0, 0)
7	6, 8, 10	(0, 1, 1)
8	7, 9, 10	(1, 1, 2)
9	6, 8, 10	(0, 1, 1)
10	7, 8, 9	(1, 1, 2)

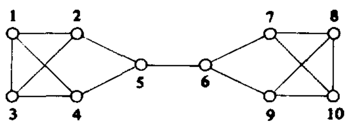

Fig. 3.

Based upon the first-order neighborhoods we obtain the partition {1, 3, 8, 10}, {2, 4, 7, 9} and {5, 6}. These are in fact the orbits of the graph and consideration of higher-order neighborhoods can only confirm the result. The use of degree vectors of neighborhoods gives some insight into the sociological structure of the network. The length of the first-order neighborhood degree vector is obviously a simple measure of point centrality. However, the content of the vector gives an indication of the extent to which each ego is contained in a clique. A high proportion of larger values (close to the vector length) would indicate a clique structure, whereas lower values would indicate contacts with no relationship with each other. As we increase the order of the neighborhoods we obtain information about closeness. The longer vectors in the higher-order neighborhoods show that ego has more contacts at a certain distance than other actors, the interval values giving some indication of whether these actors fall into cliques within contact range.

0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0	1
1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0
0	1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1
1	0	1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0
0	0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0
0	0	0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0
0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1
1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0
1	1	0	0	0	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0
0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	1	0
1	0	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1
1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	0
0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1
0	0	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	1	1	0	0
0	0	0	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	1	1	0
1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	1	1
0	1	0	0	0	1	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1
0	0	1	0	0	0	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	1
0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	1	0	0	1	0	0	0	1	0	0	1
0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	1	0	0	1	0	0	0	1
0	1	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	1	1	1	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	0	0

Fig. 4.

The techniques above are successful for most graphs but it is still possible but quite difficult to find examples on which these methods fail. Consider, for example, the graph given by the adjacency matrix in Figure 4. The graph is regular of degree 10 and has diameter 2. Hence for each x, $N(x)$ consists of 10 vertices and $\tilde{N}^2(x)$ will consist of the whole graph minus x. It turns out that $d(\tilde{N}^1(x)) =$ (3, 3, 3, 3, 3, 3, 3, 3, 3, 3) for all x, and since the graph is regular it follows that

$$d(\tilde{N}^i(x)) = d(\tilde{N}^i(y)) \quad \text{for all } x,\ y \in V \text{ and } i \geq 0.$$

However, it is the case that the graph has to orbits, $\{1..13\}$ and $\{14..26\}$. A refinement of the above technique can be incorporated into the algorithm to detect the orbits of the graph in Figure 4. Based upon one example any number of properties of the vertices could be used to distinguish between sets of similar vertices. After considering several alternatives we have decided upon using betweenness centrality (Freeman 1979) to refine the algorithm. There are several reasons for this choice; firstly betweenness has an easy sociological interpretation and most social scientists see it as a valid reason for distinguishing between individuals. Secondly, it captures "a lot of the structure" — in essence graphs in which everybody has the same betweenness measure have a high degree of combinatorial regularity which usually (but not always) means that the vertices are all automorphically equivalent. Finally it is straightforward to compute and hence easy to incorporate into the algorithm.

In a similar manner to the degree vector of Section 2 we define the betweenness vector $C_B(G)$ as follows:

$$C_B(G) = (C_B(v_1), C_B(v_2), \ldots, C_B(v_n)) \quad \text{for } v_i \in V(G)$$

where

$$C_B(v_i) \leq C_B(v_{i+1}), \quad \text{for } i = 1..n-1.$$

In the graph of Figure 4

$$C_B(\tilde{N}^1(1)) = (3, 3, 3, 3, 4, 4, 4, 5, 5, 5)$$

whereas

$$C_B(\tilde{N}^1(26)) = (3, 3, 3, 4, 4, 4, 4, 4, 4, 6).$$

The betweenness vector can be used to separate the vertex set into two subsets, namely {1..13} and {14..26}. As noted above, these correspond to the orbits of the graph.

The combination of the above concepts gives the criteria for partitioning in the algorithm.

In summary, two vertices, $x,\ y \in V$ belong to the same set provided

$$\text{(i)}\quad d(\tilde{N}^i(x)) = d(\tilde{N}^i(y)), \qquad i = 1..n-1$$

$$\text{(ii)}\quad C_{\mathrm{B}}(\tilde{N}^i(x)) = C_{\mathrm{B}}(\tilde{N}^i(y)), \qquad i = 1..n-1.$$

In other words, two vertices belong to the same set if their neighborhoods are similar with respect to the degrees and centralities of their respective elements.

4. Non-equivalence

The methods described above are intended only to detect the equivalence or non-equivalence of a pair of points; they do not attempt to measure the extent of equivalence. However, simple measurements along these lines may be devised. One such approach is given here.

First define a function SIM(i, x, y) that compares the degree and betweenness vectors of actors x and y for each neighborhood level i. If identical the function returns 1; otherwise 0.Technically,

$$\mathrm{SIM}(i,\ x,\ y) = \begin{cases} 1 & \text{if } d(\tilde{N}^i(x)) = d(\tilde{N}^i(y)) \\ & \text{and } C_{\mathrm{B}}(\tilde{N}^i(x)) = C_{\mathrm{B}}(\tilde{N}^i(y)) \\ 0 & \text{otherwise.} \end{cases}$$

We define the extent of equivalence between actors x and y, $A(x,\ y)$, as the proportion of identical neighborhoods, weighted inversely by the remoteness of the neighborhood. Specifically,

$$A(x,\ y) = \frac{\sum \mathrm{SIM}(i,\ x,\ y)/i}{\sum 1/i}$$

where the summations are over $i = 1..n-1$. Thus, if a pair of actors differs in their immediate neighborhoods ($i = 1$), they will be assigned an equivalence score close to zero. On the other hand, if they differ only in the ($n - 1$)th neighborhood, they will be assigned an equivalence score close to one. Only if they are identical on all neighborhoods will they receive a perfect score of 1.0.

Finer measures may be obtained by modifying the SIM function to return a range of values between 0 and 1 reflecting the extent of similarity between actors' degree and centrality vectors. In our current implementation (Borgatti 1987), we compute the similarity between a pair of degree vectors as the proportion of identical elements. Likewise, the similarity between centrality vectors is the proportion of values in common. If two attribute vectors are of different length, the maximum similarity between them is the length of the shorter one divided by the length of the longer one.

To accommodate the computation of extents of equivalence, we can write the algorithm as follows:

```
Initialize equivalence matrix $A(x, y) = 0$ for all $x$, $y$.
For each neighborhood level $i = 1..n - 1$ do:
for each actor $x = 1$ to $n$ do:

step 1: generate neighborhood $\tilde{N}^i(x)$
step 2: record degree vector $d(\tilde{N}^i(x))$
step 3: compute centrality vector $C_B(\tilde{N}^i(x))$

for each pair of actors $x = 2$ to $n$ and $y = 1$ to $x - 1$ do:
$A(x, y) = A(x, y) + \mathrm{SIM}(x, y)/i$.
Normalize $A(x, y) = A(x, y)/\Sigma(1/i)$.
```

One additional refinement should be mentioned. Betweenness centrality is obtained by summing the columns of the dependency matrix (Freeman (1979), just as closeness centrality is obtained by summing the rows of the geodesic distance matrix. In the summation, both measures lose information. For our purposes it is desirable that this does not occur. One way to prevent the loss is not compute the sums, but rather compare the entire column (or row) of values directly. This means that our centrality vector $C_B(\tilde{N}^i(x))$ is no longer a vector of numbers, but a vector of vectors (*i.e.* a matrix). This modification poses no serious difficulty, except that the computer implementation runs more slowly. Consequently, in our computer program we allow the user four options for computing centrality. These are:

Method	Description	Computation speed
CLOSENESS	standard closeness	very quick
CLOSENESS/V	vector of vectors	quick
BETWEENNESS	standard of betweenness	slow
BETWEENNESS/V	vector of vectors	even slower

Applying the CLOSENESS method to the graph in Figure 3, we obtain the following results:

	1	2	3	4	5	6	7	8	9	10
1	1.00	0.24	1.00	0.24	0.12	0.12	0.24	1.00	0.24	1.00
2	0.24	1.00	0.24	1.00	0.32	0.32	1.00	0.24	1.00	0.24
3	1.00	0.24	1.00	0.24	0.12	0.12	0.24	1.00	0.24	1.00
4	0.24	1.00	0.24	1.00	0.32	0.32	1.00	0.24	1.00	0.24
5	0.12	0.32	0.12	0.32	1.00	1.00	0.32	0.12	0.32	0.12
6	0.12	0.32	0.12	0.32	1.00	1.00	0.32	0.12	0.32	0.12
7	0.24	1.00	0.24	1.00	0.32	0.32	1.00	0.24	1.00	0.24
8	1.00	0.24	1.00	0.24	0.12	0.12	0.24	1.00	0.24	1.00
9	0.24	1.00	0.24	1.00	0.32	0.32	1.00	0.24	1.00	0.24
10	1.00	0.24	1.00	0.24	0.12	0.12	0.24	1.00	0.24	1.00

Two points about the equivalence matrix should be noted. First, we can confirm the fact that clustering together maximally equivalence points ($A(x, y) = 1.0$) correctly yields the orbits, which are:

$A = \{1\ 2\ 8\ 10\}$

$B = \{2\ 4\ 7\ 9\}$

$C = \{5\ 6\}$.

Second, the degree of equivalence between members of different equivalence classes corresponds to what we would intuitively expect. To see it is helpful to collapse the equivalence matrix according to the orbits, giving:

	A	*B*	*C*
A	1.00	0.24	0.12
B		1.00	0.32
C			1.00

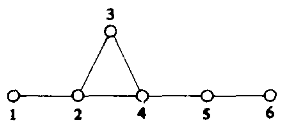

Fig. 5.

The lowest role-similarity occurs between members of A (the least central points) and members of C (the most central points). The highest similarity occurs between B and C, corresponding to the fact that the centralities of B's members are closer to those of C's members than to those of A's members.

As a further example, consider the graph in Figure 5, which has six points in six orbits. The equivalence matrix $A(x, y)$ is

	1	2	3	4	5	6
1	1.00	0.23	0.11	0.21	0.43	0.69
2	0.23	1.00	0.30	0.64	0.42	0.23
3	0.11	0.30	1.00	0.29	0.14	0.13
4	0.21	0.64	0.29	1.00	0.21	0.30
5	0.43	0.42	0.14	0.21	1.00	0.31
6	0.69	0.23	0.13	0.30	0.31	1.00

Note that the algorithm correctly identifies each pair of points as non-equivalent. Furthermore, the two most similar pairs are (1, 6) and (2, 4), as one would expect. The least similar pairs are (1, 3), (5, 3) and (6, 3), which all involve point 3, also as expected.

5. Directed graphs and networks

Both the discrete and valued forms of the algorithm may also be extended to digraphs. The definition of orbit is the same as in Section 2. We can define both indegree- and outdegree-neighborhoods as follows:

$$N_{\text{in}}(S) = S \cup \{v \in V - S\colon (v, s) \in E,\ s \in S\}$$

$$N_{\text{out}}(S) = S \cup \{v \in V - S\colon (s, v) \in E,\ s \in S\}.$$

The higher-order neighborhoods, as well as point-deleted neighborhoods, follow as before. Similarly, we define the indegree vector d_{in} as

$$d_{\text{in}}(G) = (d_{\text{in}}(v_1),\ d_{\text{in}}(v_2), \dots, d_{\text{in}}(v_n)), \quad v_i \in V$$

where $d_{\text{in}}(v_i)$ is the indegree of vertex v_i. The outdegree vector, as well as incentrality and outcentrality vectors, are defined similarly. The algorithm then checks if

1. $d_{\text{in}}(\tilde{N}^i_{\text{in}}(x)) \quad = d_{\text{in}}(\tilde{N}^i_{\text{in}}(y))$
2. $d_{\text{in}}(\tilde{N}^i_{\text{out}}(x)) \quad = d_{\text{in}}(\tilde{N}^i_{\text{out}}(y))$
3. $d_{\text{out}}(\tilde{N}^i_{\text{in}}(x)) \quad = d_{\text{out}}(\tilde{N}^i_{\text{in}}(y))$
4. $d_{\text{out}}(\tilde{N}^i_{\text{out}}(x)) \quad = d_{\text{out}}(\tilde{N}^i_{\text{out}}(y))$
5. $C_{\text{B}}(\tilde{N}^i_{\text{in}}(x)) \quad = C_{\text{B}}(\tilde{N}^i_{\text{in}}(y))$
6. $C_{\text{B}}(\tilde{N}^i_{\text{out}}(x)) \quad = C_{\text{B}}(\tilde{N}_{\text{out}}(y))$

and x and y are in the same subset if all six conditions are true for all neighborhoods.

Extending orbits to networks is simply a matter of insisting that the conditions above hold for each relation. Consequently it is straightforward to extend all the work above to networks. For the extent of equivalence algorithm we simply make the comparisons between degree and centrality vectors for each relation for each neighborhood. An overall measure of similarity can then easily be constructed.

6. Conclusion and discussion

The algorithm described above is an aid to finding the orbits of a graph. It should be emphasized that the orbits can only be finer partitions than those found by the algorithm, so that once two vertices have been separated we can guarantee that they are in separate orbits. The converse is not true; two non-separated vertices may still be in different orbits. A user analyzing social networks should at least take comfort from the fact that it is a hard task to defeat the algorithm. Sophisticated mathematical techniques are required to construct examples of graphs on which the algorithm fails. The chances of this occurring in real data — including data in which regularity is imposed upon it (*e.g.* kinship systems, organizational structures) — is so remote that it should be discounted. It is however worth noting that the procedure can fail in two distinct ways. Basically, the program examines all the neighborhoods of the vertices and concludes that vertices which have every neighborhood isomorphic are in the same orbit. Unfortunately it is computationally infeasible to test for isomorphism and so we simply examine a set of attributes of each neighborhood. Failure can occur if two non-isomorphic neighborhoods produce the same attribute measurements. A second and more dramatic failure is if two non-automorphic vertices have all their neighborhoods isomorphic. An example of a graph in this second category is the (3, 12) cage. [1] This graph has 126 vertices and two orbits; it is also regular of degree 3 and its smallest cycle is of length 12. A consequence of these facts is that every neighborhood (except when it is the whole graph) is a tree and since the graph is regular these must be isomorphic. The detailed construction of this graph is fairly complicated and the interested reader is referred to Biggs (1974: 164). The only major consequence of these possible failures is to the user genuinely looking for orbits. Since we have not characterized the graphs on which failure can occur then in this case the output of the algorithm can only be used to eliminate certain branches of a direct search.

On the other hand, the authors believe that the algorithms described above will be extremely useful to anyone interested in analyzing role equivalences in social networks.

References

Biggs, N.
1974 *Algebraic Graph Theory*. Cambridge: The University Press.
Borgatti, S.B.
1987 *Users Manual For AL: Analytic Language*. Irvine: University of California.
Colbourn, C.J. and K.S. Booth
1981 "Linear time automorphism algorithms for trees, interval graphs, and planar graphs." *SIAM Journal of Computing 10*: 203–225.
Everett, M.G.
1985 "Role similarity and complexity in social networks." *Social Networks 7*: 353–359.
Fontet, M.
1975 "Automorphismes des graphes et planarité." *Journées Algorithmiques (Ecole Norm. Sup., Paris)* Soc. Math. France: 73–90.
1976 "Linear algorithms or testing isomorphism of planar graphs," in: *Proceedings of the Third International Colloquia on Automata, Languages, and Programming*. Edinburgh University Press: 411–424.

[1] The authors are grateful to N. Biggs for pointing out this counter-example.

Freeman, L.C.
1979 "Centrality in social networks: I — Conceptual clarification." *Social Networks 1*: 215–239.

Lorrain, F. and H.C. White
1971 "Structural equivalence of individuals in social networks." *Journal of Mathematical Sociology 1*: 49–80.

Sailer, L.D.
1978 "Structural equivalence: meaning and definition, computation and application." *Social Networks 1*: 73–90.

White, D.R. and K.P. Reitz
1983 "Graph and semigroup homomorphisms on networks and relations." *Social Networks 5*: 143–234.

Winship, C. and M. Mandel
1983 "Roles and positions: a critique and extension of the blockmodelling approach," in: Leinhardt, S. (ed.) *Sociological Methodology*, pp. 316–346. San Francisco: Jossey-Bass.