

Two algorithms for computing regular equivalence

Stephen P. Borgatti

University of South Carolina, Columbia, SC, USA

Martin G. Everett

School of Mathematics, Statistics and Computing, Thames Polytechnic, London, SE18 6PF, UK

In this paper we present two algorithms for computing the extent of regular equivalence among pairs of nodes in a network. The first algorithm, REGE, is well known, but has not previously been described in the literature. The second algorithm, CATREGE, is new. Whereas REGE is applicable to quantitative data, CATREGE is used for categorical data. For binary data, either algorithm may be used, though the CATREGE algorithm is significantly faster and its output similarity coefficients have better metric properties. The CATREGE algorithm is also useful pedagogically, because it is easier to grasp.

1. Introduction

White and Reitz (1983) introduced *regular equivalence* as a formal model of the sociological notion of role. Regular equivalence represents a significant advance over *structural equivalence* (Lorrain and White 1971) in capturing key features of the relational role concept (Nadel 1957, Merton 1959). White and Reitz define regular equivalence for single-relation networks as follows:

Definition 1. If $G = \langle P, R \rangle$ and \equiv is an equivalence relation on P then \equiv is a regular equivalence if and only if for all $a, b, c \in P$, $a \equiv b$ implies:

- (i) aRc implies there exists $d \in P$ such that bRd and $d \equiv c$; and
- (ii) cRa implies there exists $d \in P$ such that dRb and $d \equiv c$.

Correspondence to: S.P. Borgatti, Department of Sociology, University of South Carolina, Columbia, SC 29208, USA.

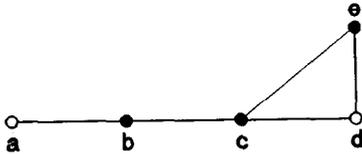


Fig. 1.

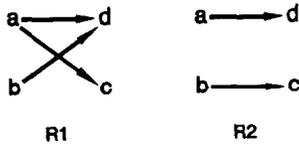


Fig. 2.

Regular equivalence may be understood as a partition of nodes into classes such that nodes of the same class are surrounded by the same classes of nodes. Figure 1 demonstrates a two-class regular equivalence. In the figure, the nodes $\{b, c, e\}$ form one equivalence class and $\{a, d\}$ form the other. Each node in the first class is connected to both a member of its own class, and a member of the other class. In contrast, each node in the second class is connected only to a member of the first class, not to a member of its own class. Thus, each class is homogeneous with respect to the kinds of nodes that its members are adjacent to. In this sense, the definition requires that equivalent actors¹ have equivalent sets of alters.

A graph may contain several distinct regular equivalences, and the set of all regular equivalences of a graph forms a lattice (Borgatti and Everett 1989). The supremum element of the lattice is known as the maximal regular equivalence or MRE. For undirected graphs with no isolates, the MRE is trivial and rarely used since the only equivalence class is the set of all nodes. In contrast, for directed graphs, the MRE is typically the most useful equivalence. Other well-known regular equivalences include *automorphic* equivalence (Everett 1985) and *structural* equivalence (Lorrain and White 1971).

¹ We use the terms 'actors', 'alter', 'node' and 'point' synonymously, but not interchangeably. Node and point are used in the context of abstract graphs, while actor is used in the context of concrete social networks. Alter refers to a node adjacent to a previously referenced node.

For networks composed of multiple relations, White and Reitz offer several approaches. The simplest approach (RE) requires only that the equivalence be regular across every relation in the network. Thus, for the network in Fig. 2, the partition $\{\{a, b\}, \{c, d\}\}$ forms a regular equivalence on each relation. Another approach suggested by White and Reitz (see also Everett and Borgatti 1992), which we shall refer to as a *multiplex regular equivalence* (MPXRE),² requires that equivalent actors have the same ‘bundle’ of relations with equivalent others. In other words, if node a has an outgoing arc with node c on relations 2 and 7 in a 10-relation network, then any node equivalent to a must have an alter equivalent to c with whom it is connected on those (and only those) two relations. Both the RE and MPXRE definitions yield a lattice of valid equivalences for any given graph, and all elements of the MPXRE lattice are also elements of the RE lattice.

For the graph in Fig. 2, the maximal MPXRE is the identity partition in which each node is in an equivalence class by itself. To see why, consider that node a is the only one that has an outgoing arc to an alter (d) on both relations. Likewise, b is the only node that has an outgoing arc to an alter (c) only on the second relation. Thus, a and b are not MPXRE equivalent, and therefore c and d are not equivalent either.

From a computational point of view, the two regular equivalence definitions are functions that take data relations (i.e. a network) as input and return *sets* of partitions (or equivalences) as output. We denote the application of the functions to a set of data consisting of relations R_i ($1 \leq i \leq p$) by $\text{RE}(R_1, R_1 \dots, R_p)$ and $\text{MPXRE}(R_1, R_2 \dots, R_p)$, respectively. Note that $\text{MPXRE}(R_1, R_2 \dots, R_p) \subseteq \text{RE}(R_1, R_2 \dots, R_p)$. Furthermore, the MPXRE function applied to a set of relations is equivalent to the RE function applied to the same set of relations plus their intersections. For example, $\text{MPXRE}(R_1, R_2, R_3) = \text{RE}(R_1, R_2, R_3, R_1 \cap R_2, R_1 \cap R_3, R_2 \cap R_3, R_2 \cap R_3, R_1 \cap R_2 \cap R_3)$.

Of special interest is the case where R_1 is any relation R and R_2 is the inverse R^{-1} . Obviously, $\text{RE}(R) = \text{RE}(R, R^{-1})$, since regular equivalence takes both incoming and outgoing ties into account.

² Multiplex Regular Equivalence is another name for the ‘bundle equivalence’ that White and Reitz (1983: 208) introduce. We do not use their term because in the same article (214) they use ‘bundle equivalence’ to refer to the distinctly different work of Mandel and Winship (1979).

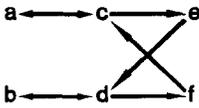


Fig. 3.

However, $\text{MPXRE}(R) \neq \text{MPXRE}(R, R^{-1})$. The equivalences generated by $\text{MPXRE}(R, R^{-1})$ have the following very useful property: if two actors are equivalent, they must have the same combination of incoming and outgoing arcs with equivalent alters. This is readily understood by considering the case where R_2 is *not* the inverse of R_1 . For example, if we consider the ‘works with’ and ‘lends money to’ relations and suppose that actors c , d , and e have been found equivalent, and actor a who both lends money and works with an actor c will not be considered equivalent to an actor who lends money to (but does not work with) d even if she works with (but does not lend money to) e . In other words, lending money to a coworker is considered a unique social relation that affects one differently from the experience of lending money to non-coworkers and from the experience of working with someone who is not indebted to you. Thus, returning to the case of $\text{MPXRE}(R, R^{-1})$, all equivalences generated by this function will have the property that a reciprocated relationship between two nodes will be treated differently from separate outgoing and incoming links.

2. The REGE algorithm

The REGE algorithm is the result of efforts of a group of University of California, Irvine researchers, including Lee Sailer, John P. Boyd, Douglas R. White and Karl Reitz. The essential points of the algorithm were presented by D.R. White in three unpublished papers (1980, 1982, 1984). The earliest computer implementation known to the present authors is a FORTRAN program from 1985 by D.R. White. This was translated into BASIC by L.C. Freeman (1985) as part of the UCINET 2.0 computer package, and rewritten in BASIC by Bruce MacEvoy and L.C. Freeman (1987) as part of the UCINET 3.0 package. More recently, the program has been translated into PASCAL as part of the UCINET IV package (Borgatti *et al.* 1992a).

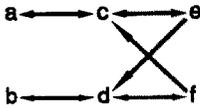


Fig. 4.

While simple in many respects, there are certain aspects of REGE's operation that are not altogether clear. For example, many practitioners (including ourselves) have assumed that REGE locates that MRE. However, this is not the case, as the graph in Fig. 3 illustrates. For that graph, the REGE algorithm identifies the following partition: $\{\{a\ b\} \{c\ d\} \{e\ f\}\}$. Yet the MRE partition is $\{\{a\ b\ c\ d\ e\ f\}\}$ (all nodes equivalent) since the graph contains no sinks or sources (Borgatti and Everett 1989).³ Based on the pattern of arcs in the graph one might surmise that REGE distinguishes actors who have both incoming and outgoing links with a single alter from actors who have outgoing links with one set of alters and, separately, incoming links with another set of alters. Such a distinction would mean that REGE was computing the maximal $MPXRE(R, R^{-1})$, which would be quite sensible and desirable. However, this is also not the case, as illustrated by the graph in Fig. 4. Here, REGE returns the complete partition $\{\{a, b, c, d, e, f\}\}$, despite the fact that e and f are the only nodes with unreciprocated arcs, making $\{\{a, b\}, \{c, d\}, \{e, f\}\}$ the maximal $MPXRE(R, R^{-1})$.

REGE is an iterative algorithm that yields a measure e_{ij} of the extent of equivalence (*i.e.* role similarity) of all pairs of nodes i and j . It begins by setting $e_{ij} = 1.0$ for all pairs of nodes. With each succeeding iteration, it recomputes e_{ij} for all pairs based on the degree to which i 's alters correspond to j 's alters.

For the first iteration, e_{ij} is calculated by counting up the extent to which i 's ties to her alters correspond to j 's ties to his alters (and vice versa), and dividing by the total possible. Both the numerator and denominator are influenced by the sheer number of alters each actor has. The extent of correspondence is computed on a point system as follows. Consider the disconnected graph in Fig. 5. The neighborhood

³ A source is a node with outdegree, but no indegree. A sink is a node with indegree, but no outdegree. For a fuller discussion of the importance of sinks and sources for regular equivalence, see Borgatti and Everett (1989).

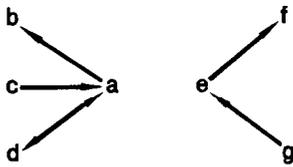


Fig. 5.

(or ‘set of alters’, which we denote by N) of node a is $N(a) = \{b, c, d\}$. Note that the link with b is outgoing only, the link with c is incoming only, and the link with d is both incoming and outgoing. The neighborhood of node e is $N(e) = \{f, g\}$. Node f constitutes an outgoing link for e whereas g gives an incoming link. Suppose we are computing the extent of equivalence between nodes a and e . The algorithm begins by seeking elements in e ’s neighborhood that match a ’s. Since the relationship (an outgoing tie) between a and b is matched by the relationship between e and f , a point is scored. Similarly, a ’s relationship with c is matched by e ’s relationship with g in $N(e)$, which scores another equivalence point.

Next we consider alter d in $N(a)$, which constitutes both an incoming and an outgoing link for a . There are several ways an algorithm could handle this situation. One possibility is to let d match only an alter in $N(e)$ which is both an incoming and an outgoing link. This choice yields the MPXRE partition (see Table 1). In this example, this would mean that d has no match in $N(e)$, and therefore the extent of equivalence between a and e at this stage in the computation is $2/3$.

Another possibility is to treat alters such as d as wildcards which match any kind of alter, since they represent both incoming and outgoing arcs. This choice yields the RE partition (see Table 2). In the

Table 1
MPXRE(R, R^{-1}) point system for evaluating equivalence between actors a and e

		Alter j		
		$e \rightarrow j$	$e \leftarrow j$	$e \leftrightarrow j$
Alter i	$a \rightarrow i$	1	0	0
	$a \leftarrow i$	0	1	0
	$a \leftrightarrow i$	0	0	1

Table 2
RE point system for evaluating equivalence between actors a and e

		Alter j		
		$a \rightarrow j$	$e \leftarrow j$	$e \leftrightarrow j$
Alter i	$a \rightarrow i$	1	0	1
	$a \leftarrow i$	0	1	1
	$a \leftrightarrow i$	1	1	1

Table 3
REGE point system for evaluating equivalence between actors a and e

		Alter j		
		$e \rightarrow j$	$e \leftarrow j$	$e \leftrightarrow j$
Alter i	$a \rightarrow i$	1	0	1
	$a \leftarrow i$	0	1	1
	$a \leftrightarrow i$	1	1	2

example, d would then match either f or g in $N(e)$, for a (partial) similarity score of $3/3$.

REGE, however, chooses neither alternative. Instead, REGE implements a kind of compromise (Table 3). It allows the d to match any alter in $N(e)$, as in the wildcard approach, but if $N(e)$ contains a better match for d (an alter that was both an incoming and outgoing link), the match scores 2 points (one for each direction). In effect, REGE counts the number of links matched rather than the number of alters. The result is an equivalence that is neither MRE nor MPXRE in general, but can be either on occasion.

Once it has counted matches from one node's point of view, REGE repeats the process from the other node's point of view. In the example, it would start with f and seek a match in $N(a)$. Since b (or d) matches, a point is scored. Then it seeks a match for g . This time c (or d) matches, and another point is scored. Points from both nodes of view are summed and the result divided by the maximum possible. For this example, $REGE(R)$ would compute:

$$\frac{(1 + 1 + 1) + (1 + 1)}{(1 + 1 + 2) + (1 + 1)} = 5/6$$

In contrast, an RE(R) algorithm would compute:

$$\frac{(1 + 1 + 1) + (1 + 1)}{(1 + 1 + 1) + (1 + 1)} = 5/5 = 1$$

and an MPXRE(R, R^{-1}) algorithm would compute

$$\frac{(1 + 1 + 0) + (1 + 1)}{(1 + 1 + 1) + (1 + 1)} = 4/5$$

which, in this particular case, resembles the REGE result.

On the second, iteration, matching occurs the same way, but now the points counts are weighted by the previous iteration's equivalence between the matched alters. For example, when matching a 's b with e 's f , we multiply the points from the match (totalling 1) by the extent of equivalence between b and f computed in the previous iteration. Since that number must be 1 or less, the outcome of the multiplication is less than or equal to one. Thus the terms in the numerator get knocked down with each iteration, but the terms in the denominator do not. Hence the equivalence scores for each iteration diminish (for non-equivalent pairs) with each iteration.⁴

How, then, do we interpret the similarity scores at the end of k iterations? Tracing the results of each iteration for simple graphs reveals that the first iteration assigns zero similarity to all pairs of nodes involving just one isolate,⁵ and assigns low similarity to all pairs of nodes in which just one node is a sink or a source. The next iteration further reduces the similarity of pairs containing sinks and sources, and begins reducing the similarity of pairs in which one node's neighborhood contains a source or sink and the other does not.

The net result of the process is to progressively reduce the similarity between pairs based on the distance that each of the two nodes is from every sink and source in the graph.

Pairs of nodes that have a different distribution of distances from each sink and source will be assigned progressively smaller similarity.

⁴ The rate of attenuation is not a constant, nor is it the same for different pairs of nodes.

⁵ Most implementations artificially assign perfect similarity to pairs in which both actors are isolates.

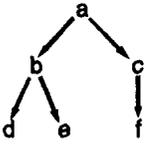


Fig. 6. Graph with REGE partition $\{\{a\}, \{b, c\}, \{d, e, f\}\}$.

Table 4
REGE output (3 iterations) based on graph in Fig. 6

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>a</i>	100	40	34	0	0	0
<i>b</i>	40	100	100	25	25	23
<i>c</i>	34	100	100	33	33	31
<i>d</i>	0	25	33	100	100	100
<i>e</i>	0	25	33	100	100	100
<i>f</i>	0	23	31	100	100	100

The actual quantities, however, will depend upon the number of alters in each nodes's neighborhood, and on the proportion of bidirectional links they have. These dependencies make similarity values which are not 1 or 0 difficult to interpret with precision. Further, it is known that the values between iterations for any given dataset need not be highly correlated, even at the rank-order level.⁶

Another problem concerns the metric properties of REGE's measure of the extent of equivalence. The graph in Fig. 6 has REGE equivalence classes $\{a\}$, $\{b, c\}$, $\{d, e, f\}$. The matrix of similarity measures is given in Table 4. Consider the extent of equivalence between node *a* in the first class and nodes *b* and *c* in the second. Nodes *b* and *c* are perfectly equivalent, yet they are not equally equivalent to *a*. Thus, REGE's measure of similarity is not (isomorphic with) a distance metric. REGE's metric properties can fail whenever otherwise equivalent nodes have different degree. Since regular equivalence as a mathematical concept is insensitive to degree, this represents a serious gap between the ideal type and the algorithmic instantiation.

⁶ It is customary to stop the program after three iterations, but this arbitrary.

3. The CATREGE algorithm

In this section we present an alternative algorithm (see Fig. 7) for computing the maximal MPXRE(R, R^{-1}). Just as the REGE algorithm is easily modified to yield an RE or MPXRE solution, so is the CATREGE algorithm. Since we have chosen the MPXRE(R, R^{-1}), however, we have been able to introduce certain computational efficiencies. For example, the presence of R^{-1} in the data obviates the need for the second (or 'indegree') condition of Definition 1. Hence, the CATREGE algorithm performs only the first (or 'outdegree') check.

The algorithm takes a *multiplex* matrix (White and Reitz 1983: 208; Borgatti *et al.* 1992b: 55) as input. The values of a multiplex matrix X are categorical codes that index each unique combination of input relations (and their inverses) that connect each pair of nodes. For example, if the data consist of a single directed relation, the possible values of x_{ij} are 1 (if iRj but not jRi), 2 (if jRi but not iRj), 3 (if iRj and jRi) and 0 (if not iRj and not jRi). An algorithm for computing X from a collection of one or more input relations is given in Fig. 8.

Given a multiplex matrix (or any other categorically valued matrix), the CATREGE algorithm explodes the input data into as many binary relations as there are distinct values, such that $(i, j) \in R_k$ iff $x_{ij} = k$. Then for each of these derived relations, the algorithm iteratively verifies that pairs of nodes that were equivalent in the previous iteration have the same types (classes) of nodes in their neighborhoods. If so, the nodes remain equivalent; if not, they are marked as non-equivalent and are not considered in subsequent iterations.

All nodes are assumed equivalent prior to the first iteration. It is this assumption that selects the maximal regular equivalence consistent with the input relations. It is important to note, however, that other regular equivalences may be selected by choosing different starting assumptions. For example, to compute the most inclusive regular equivalence that is also consistent with a given node-attribute (e.g. centrality), we would first partition nodes by that attribute (i.e. two nodes are in the same class if they have the same score on the attribute of interest) and use that as the starting partition. This is useful if the equivalence classes one seeks need to be homogeneous with respect to a particular attribute in order to be theoretically useful. An example might be the case where regular equivalence is

```

Const
  maxn = 255; {maximum number of nodes}
  maxr = 10; {maximum number of distinct relational bundles}
Type
  bytevector = array[1..maxn] of byte;
  byteset    = set of byte;
  matrix     = array[1..maxn] of bytevector;

Procedure CATREG (var mpx:matrix; n,r:byte);
{ mpx is input categorically valued matrix      }
{ p is output matrix of partitions, one per row }
{ n is the number of nodes; r is the number of distinct bundles }
Var
  it,i,j:      integer;
  changes:     boolean;
  part1,part2: bytevector;
  nb:         array[1..maxr,1..maxn] of byteset;

  Procedure ComputeNeighborhoods;
  Var
    i,j,q: byte;
  Begin
    for q:= 1 to r do for i:= 1 to n do nb[q,i]:= [];
    for i:= 1 to n do for j:= 1 to n do begin
      q:= mpx[i,j];
      if q > 0 then nb[q,i]:= nb[q,i] + [part1[j]];
    end;
  End;

  Function Same(i,j: byte): boolean;
  Var
    q: byte;
  Begin
    same:= false;
    for q:= 1 to r do if nb[q,i] <> nb[q,j] then exit;
    same:= true;
  End;

Begin { catreg }
  it:= 0;
  for i:= 1 to n do part1[i]:= 1;
  repeat
    computeneighborhoods; inc(it);
    p[it]:= part1;
    changes:= false;
    for i:= 1 to n do part2[i]:= i;
    for i:= 2 to n do for j:= 1 to i-1 do
      if part1[i] = part1[j] then if same(i,j)
        then part2[i]:= part2[j]
        else changes:= true;
    part1:= part2;
  until not changes;
End;

```

Fig. 7. CATREG algorithm for computing $MPXRE(R, R^{-1})$. Assumes multiplex matrix as input.

Table 6
CATREGGE similarities based on the graph in Fig. 9.

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>
<i>a</i>	5	1	1	1	1	1	1	1	1
<i>b</i>	1	5	2	2	2	2	2	2	1
<i>c</i>	1	2	5	3	3	3	3	2	1
<i>d</i>	1	2	3	5	4	4	3	2	1
<i>e</i>	1	2	3	4	5	4	3	2	1
<i>f</i>	1	2	3	4	4	5	3	2	1
<i>g</i>	1	2	3	3	3	3	5	2	1
<i>h</i>	1	2	2	2	2	2	2	5	1
<i>i</i>	1	1	1	1	1	1	1	1	5

The output of CATREGGE is a collection of hierarchically nested partitions. In all cases, the first partition has all nodes in the same equivalence class (unless a different starting partition was used). Succeeding partitions break up the classes of the previous partition into smaller classes whose members are 'more' equivalent. The classes of the last partition contain only nodes which are perfectly regularly equivalent (in many cases, the final classes will be singletons, each containing a single node). This hierarchical set of partitions can be displayed as a cluster diagram, as shown in Table 5.

A natural measure of the extent of regular equivalence between a pair of nodes is given by the number of iterations needed to split them into separate classes. If they are split after the first iteration, it means that they have grossly different relational patterns, since the first iteration essentially splits up the nodes according to whether they are sinks, sources, or repeaters. If they are split only after the second iteration, it means that they are the same basic type (*i.e.* sink, source or repeater) but their immediate neighborhoods do not contain the same combinations of sinks, sources and repeaters. If two nodes are never split apart, then they are perfectly equivalent. A normalized measure varying between 0 and 1 can be obtained by dividing by the total number of iterations. Table 6 gives the similarities obtained from running the algorithm on the graph shown in Fig. 9.

Note that, in contrast to REGE, this measure of similarity is not



Fig. 9.

affected by the degree of nodes, nor does it yield inconsistent results (e.g. in the graph in Fig. 6, the similarity of *a* to *b* is the same as from *a* to *c*, unlike REGE's results). Furthermore, the similarity coefficients are clearly defined.

Another advantage of this algorithm is its speed. Whereas REGE runs in time proportional to n^5 , CATREGE runs in time proportional to n^3 . This allows very large networks to be processed in the same amount of time required for a single matrix multiplication. Associated with its speed is its simplicity. The algorithm is easy to comprehend and therefore valuable pedagogically.

4. Summary

Most graphs possess several regular equivalences, which form a lattice. The well-known REGE algorithm finds one of them, but it is not always clear which one. It is neither the maximal regular equivalence (MRE) nor the multiplex regular equivalence (MPXRE), though it may coincide with either. Analysis of the algorithm reveals that the uncertainty is due to the implicit point system which assigns varying points to different degrees of correspondence between actors' relationships. The point system is such that idiosyncracies of the data (such as unequal degree and unequal numbers of reciprocated relationships) can influence which equivalence is selected from the lattice. This is also the source of another problem, which is that equivalent nodes need not be equally equivalent to other nodes. These problems can be corrected by adjusting the point system, as shown in Tables 3 and 4. However, the resulting similarity measures (at each iteration) remain difficult to interpret, and the user must still arbitrarily choose which iteration to accept.

The CATREGE algorithm, in contrast, produces a single, simpler measure of similarity, without requiring an arbitrary choice of iterations. The algorithm is also orders of magnitude faster. An important limitation, however, is that CATREGE cannot properly be used with quantitative data: it treats all data values as categorical. Thus, for data in which the strength of relationships among actors has been collected, the REGE algorithm is more appropriate.⁷

⁷ The reader may also wish to consult the REGD algorithm described by Reitz and White (1989).

One problem suffered by both REGE and CATREGE is the lack of a theoretical rationale for the measure of similarity produced. While CATREGE's measure is simpler and clearer than REGE's, it cannot be claimed to be any more *valid* (assuming we make the corrections in REGE discussed above), because validity would imply a correspondence between the measure and well-defined theoretical definition of the extent of regular equivalence between two nodes. But such a definition does not exist: all we have is definitions of perfect regular equivalence. In this sense, the blockmodeling approaches to computing regular equivalence of Batagelj *et al.* (1992) and Borgatti and Everett (1992) are superior, because they do not require a definition of the 'regular similarity' of pairs of nodes. The weakness of the blockmodeling approaches, however, is that, given the current state of development of combinatorial optimization techniques, they are too slow for many applications. In this sense, the similarity-based methods like REGE and CATREGE are strong computationally but weak analytically, while the blockmodeling methods are strong analytically but weak computationally.

Both REGE and CATREGE are implemented in the UCINET IV software package (Borgatti *et al.* 1992a).

References

- Batagelj, V., P. Doreian, and A. Ferligoj
 1992 "An optimizational approach to regular equivalence." *Social Networks* 14: 121–135.
- Borgatti, S.P. and M.G. Everett
 1989 "The class of all regular equivalences: algebraic structure and computation." *Social Networks* 11: 65–88.
 1992 "Regular blockmodels of multiway, multimode matrices." *Social Networks* 14: 91–120.
- Borgatti, S.P., M.G. Everett, and L.C. Freeman
 1992a *UCINET IV Version 1.00*. Columbia: Analytic Technologies.
 1992b *UCINET IV Reference Manual*. Columbia: Analytic Technologies.
- Everett, M.G.
 1985 "Role similarity and complexity in social networks." *Social Networks* 7: 353–359.
- Everett, M.G. And S.P. Borgatti
 1993 "An extension of regular colouring of graphs to digraphs, networks and hypergraphs." *Social Networks* 15: 237–254.
- Freeman, L.C.
 1985 *UCINET 2.0 Microcomputer Package*. Irvine, CA: University of California, School of Social Sciences.
- Lorrain, F. and H.C. White.
 1971 "Structural equivalence of individuals in social networks." *Journal of Mathematical Sociology* 1: 67–80.

MacEvoy, B. and L.C. Freeman

1987 *UCINET: A Microcomputer Package for Network Analysis*. Irvine, CA: University of California, School of Social Sciences.

Mandel, M.J. and C. Winship

1979 "Roles, positions and networks." Paper presented to The American Sociological Association meetings, Boston, MA.

Merton, R.K.

1959 *Social Theory and Social Structure*. 2nd edition. Glencoe, IL: Free Press.

Nadel, S.F.

1957 *The Theory of Social Structure*. London: Cohen & West.

Reitz, K. and D.R. White

1989 "Rethinking the role concept: homomorphisms on social networks." In: Freeman, D.R. White, and A.K. Romney (Editors), *Research Methods in Social Networks Analysis*, Fairfax, VA: George Mason University Press, pp. 429–488.

White, D.R.

1980 "Structural equivalences in social networks: concepts and measurement of role structures." Paper presented at Research Methods in Social Network Analysis Conference, Laguna Beach, California, April 1980.

1982 "Measures of global role equivalence in social networks." Unpublished manuscript.

1984 "REGGE: a REGular Graph Equivalence algorithm for computing role distances prior to blockmodeling." Unpublished manuscript.

White, D.R. and K. Reitz

1983 "Graph and semigroup homomorphisms on semigroups of relations." *Social Networks* 5: 193–234.