Social Networks 15 (1993) 151–170
North-Holland

A linear algorithm for computing automorphic equivalence classes: the numerical signatures approach [1]

Malcolm K. Sparrow
Harvard University, Cambridge, MA 02138, USA

An efficient method for computing role (automorphic) equivalences in large networks is described. Numerical signatures (real numbers) are generated for each node. Role-identical nodes share common numerical signatures. The decomposition of the node set into classes by reference to the numerical signatures helps determine the automorphic equivalence classes of a network. The technique is applicable to symmetric and directed networks, and to graphs with multiple link types. The algorithm is linear with respect to the number of links in the network. Its theoretical foundation exploits properties of transcendental numbers.

1. Introduction

This paper presents an extremely simple computational approach to the determination of role equivalences within large networks. The technique produces simple numerical "signatures" for each node in such a way that role equivalent nodes will share common signatures.

The computational burden is linear with respect to the total number of links in the network. The technique applies equally well for symmetric and non-symmetric graphs, and for networks with multiple link types.

This technique is developed with the task of analyzing criminal enterprises and organizations in mind. The goal is to develop powerful computational techniques for identifying players with particular or key roles within criminal networks.

Correspondence to: M.K. Sparrow, RM L306, Kennedy School of Government, 79 John F. Kennedy Street, Cambridge, MA 02138, USA.

[1] This research was supported by a grant from the US army and the MITNE Corporation.

That purpose produces three requirements. First, the provision of computational methods for finding role equivalences which are simple enough and fast enough for practical application to very large networks. Second, examination of what it might mean in different contexts to be "close to role equivalent", especially in an environment of incomplete information (such as intelligence databases). Third, that the computational methods then be adapted to reveal role similarity as well as identity.

This paper addresses the first of these three requirements. The other two will be addressed subsequently.

The technique developed here will appeal to anyone seeking to compute role equivalence classes in large networks, whatever the field of application. Sufficient detail of the technique is presented in this paper to make implementation in any computing environment a fairly simple programming task.

2. Background

The network analysis literature contains several distinct concepts of structural equivalence.

The simplest is "substitutability" or "interchangeability", where two nodes share identical first order neighborhoods (Lorrain and White 1971; Burt 1988).

A second key concept is "stochastic equivalence", where two nodes share the same *a priori* probability of being connected to any other designated node in the network, and where those probabilities need not be zero or unity (see Wasserman and Anderson 1987, definition 4; Holland et al 1983.

A third key notion, and the subject of this discussion, is automorphic equivalence or "role equivalence". It captures the idea that nodes (actors) can play equivalent roles with respect to different sets of players, or even in entirely separate organizations. Role-equivalence classes are familiar to mathematicians as the "orbits" under the Group of permutations comprising edge-preserving automorphisms of a Graph (see, for instance, Wilson 1972, Biggs 1974).

This more sophisticated concept of equivalence (a mathematically weaker condition than substitutability) was introduced to the network analysis literature by Sailer (1978). Despite its intuitive appeal, confu-

sion between role equivalence and the other forms of equivalence persisted through the early 1980s. Role equivalence was mentioned by Doreian (1987 and 1988) as "regular equivalence" although he didn't dwell on its significance. The concept was then spelt out in great detail by Faust (1988), and very clearly by Everett and Borgatti (1988). As it is fundamental to this paper, let us define role equivalence once more:

"In a network X, a permutation of the nodes f is an automorphism if and only if it preserves adjacency: that is, given two nodes a and b, then $f(a)$ is linked to $f(b)$ if and only if a is linked to b. Two nodes are said to be role equivalent (or automorphically equivalent) if there is an automorphism which maps one onto the other. That is, a and b are automorphically equivalent if there exists an automorphism f such that $f(a) = b$."

2.1 Computational methods

Substitutability is relatively simple to compute. Substitutable nodes have identical vectors in the adjacency matrix. In non-symmetric networks (directed graphs) nodes are substitutable if and only if their respective rows are identical and their respective columns also.

It is important to note that blockmodelling discerns the "substitutability" of nodes as opposed to any other form of equivalence. See Breiger et al (1975) regarding the CONCOR algorithm. Panning (1982) develops goodness of fit measures for blockmodels. See Heil and White (1976) re BLOCKER, and Everett (1982) re EBLOC, a fast blocking algorithm based on a slightly extended form of substitutability.)

Some algorithms that assist in finding role similarities have been suggested. Doreian (1987) suggests applying REGE to symmetric networks, first decomposing them into two directed networks. Borgatti (1988) points out some severe practical difficulties with that approach. Everett and Borgatti (1988) present a method based on examining the "degree vectors" of successively higher order neighborhood Also Burt (1990) describes an approach developed by Hummell and Sodeur for detecting role equivalence by counting and classifying types of triadic relations. This method unfortunately only takes close neighborhoods into account, not any deeper structure.

3. The concept of numerical signatures

The basic concept underlying numerical signatures, applied first to symmetric networks, is as follows. Imagine that each node fires a numerical signal at time $t(0)$ along each of its links. We'll use the degree of the node (an integer) for this original signal. At time $t(1)$ each node receives the incoming signals from all adjacent nodes, and mathematically combines them to produce one single resulting (real) number. Call the result of that calculation the "1st order numerical signature" for each node.

The numerical signatures are immediately retransmitted by each node along every available link, being received by adjacent nodes at time $t(2)$. Again the incoming signals are combined to form the 2nd order numerical signatures.

The process of combining incoming signals and retransmitting them is continued to any desired depth. The signature of a node after i iterations then contains information from all the nodes reachable by paths of length not exceeding i (which Everett and Borgatti (1988) call the ith-order neighborhood of the node).

The appeal of such an approach is twofold. First, comparing numerical signatures is delightfully easy, both conceptually and computationally (much easier, for example, than comparing sequences of degree vectors as suggested by Everett and Borgatti (1988)). Second, the process of moving from the (n)th order signature to the $(n + 1)$th is no more complex, no matter how large n, than moving from depth 1 to depth 2. The computations are the same.

The challenge is to find an appropriate method of combining incoming signals to produce a single number. Suppose a node receives signals {1,1,1,4,7} at time $t(1)$ (i.e. it has degree 5, and the adjacent nodes have degrees 1,1,1,4 and 7). Then the resulting numerical signature, to be useful, has to be unique to that combination of incoming signals. It must also be unchanged by any permutation of the incoming signals (i.e. it needs to be a commutative operation).

Adding the signals together meets the second condition, but not the first. So does multiplying them. Something slightly more complex is required.

3.1 "Add π and multiply"

Adding the irrational number π (approximately 3.14159237) to each and then multiplying them together does produce the desired effect. So we define the 1st order numerical signature of node i as:

$$S_{i,1} = \prod^{k} (\pi + S_{k,0}) \tag{1}$$

where $S_{k,0}$ is the signal transmitted from node k at time $t(0)$, and the product is evaluated for all k adjacent to i. We will call π the "compounding factor" used at depth 1.

$S_{i,1}$ is therefore a finite polynomial in π, with integer coefficients. Now consider the possibility that $S_{i,1} = S_{j,1}$ for some $i \neq j$. $S_{j,1}$ is also a finite polynomial in π with integer coefficients. So consider

$$S_{i,1} - S_{j,1}$$

This must be another finite polynomial in π equal to zero. But we know that the number π is not only irrational, but has a further special mathematical property, called *transcendence*, which means that it cannot satisfy any non-trivial finite polynomial with integer coefficients. The existence of transcendental numbers was first proved by Liouville in 1840, and Lindemann was the first to prove the transcendentality of π, laying to rest in 1882 a question which had been the subject of speculation amongst mathematicians for over two thousand years (Borwein and Borwein 1987).

Therefore $S_{i,1} - S_{j,1}$ must be the zero polynomial (all coefficients zero), which means that $S_{i,1}$ and $S_{j,1}$ must be identical polynomials (i.e. with all coefficients identical).

If $S_{i,1}$ and $S_{j,1}$ are identical polynomials it follows that they were generated from the same set of incoming signals (because a polynomial is determined by its roots, and a polynomial of degree n cannot have more than n roots).

Thus the only way that $S_{i,1}$ and $S_{j,1}$ can take the same numerical value is for nodes i and j to have received identical sets of incoming signals both in size and content, but subject to permutation. And that is exactly the property we were looking for.

3.2. Greater depths

Suppose we generate 2nd order signatures using exactly the same procedure, and using the same compounding factor π in the calculations. And suppose, once more, that $S_{i,2} = S_{j,2}$ for some $i \neq j$.

It is tempting to apply the same logic as above. But to do so would overlook the fact that the received signals will no longer be integers. Nor need they be rational. They are themselves polynomials in π and are therefore irrational. So the new polynomial in π, which constitutes $S_{i,2}$ for any particular i, has irrational coefficients (each of which is some polynomial in π). So we cannot claim that this second-stage combination of signals produces a signature which is unique to the incoming set of signals at that stage.

To build the appropriate theoretical basis for 2nd and higher order signatures we need a conjecture which, fortunately, we will not have to prove for the practical purposes of this paper.

Conjecture: That there exists a sequence of transcendental numbers $\{\pi_i: i = 1,2,\ldots,n\}$ which *together* do not satisfy any non-trivial polynomial with integer coefficients. (We might perhaps call such numbers *mutually transcendental*.) In other words, there does not exist a set of non-zero integer coefficients $\{c_j\}$, and sets of positive integer powers $\{p(j), q(j), r(j),\ldots,: j = 1,2,\ldots,m\}$ such that

$$\sum_{j=1}^{m} c_j \pi_1^{p(j)} \pi_2^{q(j)} \pi_3^{r(j)} \ldots, \pi_k \ldots = 0 \tag{2}$$

Assume, for a moment, the existence of such a sequence of transcendental numbers $\{\pi_i: i = 1,2,\ldots,n\}$. Then we would have all the tools we need to extend the process to the 2nd-order signature and higher orders. At the 2nd order stage simply use π_2 as the compounding factor, rather than π_1 (assume that π_1 actually took the numerical value π). At the nth-order stage use π_n.

Now suppose that $S_{i,n} = S_{j,n}$ for some $i \neq j$. Then $S_{i,n} - S_{j,n}$ takes the value zero. But $S_{i,n} - S_{j,n}$ is a finite polynomial in π_n with coefficients each of which are finite polynomials in $\{\pi_1, \pi_2 \ldots \pi_{n-1}\}$. Thus, by the conjecture, all the coefficients have to be zero, which means that $S_{i,n}$ and $S_{j,n}$ must have been identical polynomials.

Using such a sequence of transcendental numbers guarantees that, for any value of n, if $S_{i,n} = S_{j,n}$ for some $i \neq j$, then the signals

received at node i at time $t(n)$ must have been some permutation of the signals received at node j at time $t(n)$.

In which case, if the real numbers generated as the nth order signature of node i and node j for some $i \neq j$ are the same, then all of the signals which have gone into the manufacture of those signatures throughout the entire process must have been identical.

In fact the numerical nth order signatures are remarkably close to having a 1-to-1 mapping from the possible structures of the nth order neighborhoods.

3.3. Practical implementation of the algorithm

Before this computational technique can be tested on complex networks certain practical realities have to be addressed. First, we need something resembling the conjectured list of transcendental numbers to feed the algorithm. Second, we have to face the fact that, in computing, we will necessarily be dealing with rational approximations (truncations) for irrational numbers.

The reason that the conjecture does not need to be proved here is that, for all practical computational purposes, transcendental numbers are not needed. Nor even irrational numbers: random numbers will do. It is only necessary to provide sufficient complexity to the process to eliminate numerical "flukes". Floating point arithmetic, operating with 14 significant figures, coupled with the use of n random numbers in the place of the set $\{\pi_i\colon\ i = 1,2,\ldots,n\}$, achieves the purpose perfectly well.

Nor is the author aware of any graph which actually requires different compounding factors to be used at successive stages of the process. The algorithm has been tested extensively using the same number, 3.14159237 (a truncated approximation to π) at every stage, and the algorithm has never produced the same signatures from two different sets of incoming signals. [2]

3.4. Truncation errors

One other computational issue arises. The technique assumes that multiplication of a set of numbers is truly commutative: i.e. that it

[2] It may in theory be possible to design graphs which would mandate the use of different compounding factors at successive stages.

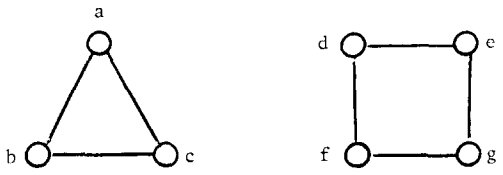

Fig. 1.

does not matter in what order they are taken. Algebraically that is true. Computationally, owing to rounding errors, it is not. The least significant digits of a signature might be affected, therefore, by the order in which the contributory signals are taken. This problem is easily resolved (without having to allow margins for error) by ordering the signals in increasing or decreasing numerical size prior to calculating the product.

3.5. Cycles

One other refinement also turns out to be desirable. When the algorithm, as described so far, is applied to Fig. 1, which consists of one cycle of length 3 and a disjoint cycle of length 4, the numerical signatures fail to separate nodes $\{a,b,c\}$ from nodes $\{d,e,f,g\}$. It puts them all in the same class, thus failing to disaggregate completely the node set into its automorphic equivalence classes.

It is easy to understand why. At stage one each node receives two signals of "2.0", as each is connected to two nodes of degree two. The 1st order signature of every node is therefore $(2+\pi)^2$. Next, each node transmits that same number so, once again, every node received two identical signals. And so on. The signatures will all be identical at every stage. The problem is that nothing in the algorithm notices that, in the case of node a at the 3rd stage, incoming signals contain information which originated at a, which has travelled around a cycle of length 3, and has returned to contribute to a's 3rd order signature. The same is true for nodes b and c, but not for d, e, f or g.

The requisite refinement is to keep track, at each stage, of the number of nodes which have contributed to any one signature. The 4th order signatures for nodes d, e, f and g contain information originating from a total of 4 nodes, whereas the 4th order signatures for a, b and c still only span 3 nodes.

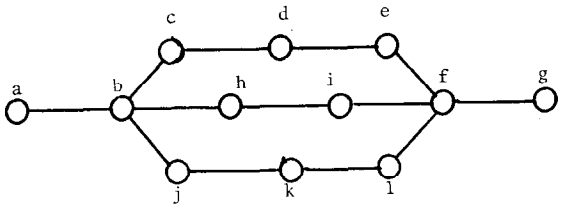

Fig. 2.

For the algorithm to discriminate successfully between nodes on the basis of cyclical structures we need to build this refinement into the computation of signatures. This is accomplished by updating a binary copy of the adjacency matrix in such a way that the jth column at time $t(n)$ shows which nodes have contributed to node j's nth order signature. This is equivalent to keeping track of the nodes reachable by paths of length $\leqslant n$ from node j.

Then, when calculating the nth order signatures, first calculate the number of contributing nodes (the size of the nth order neighborhood). Call it *SPAN*. And incorporate *SPAN* into the signature calculation in some suitably complex way — for example:

$$S_{i,n} = (SPAN + \sqrt{\pi}) * \prod^{k} (\pi + S_{k,n-1}) \tag{3}$$

It is important that *SPAN* is incorporated into the calculation in a different way from the incoming signals, lest it be confused with any of them that happen to be integers. [3]

3.6. *Application to networks*

This last refinement enables the algorithm to discern differences in cyclical structures fairly effectively. The graph in Fig. 2 has automorphic equivalence classes $\{a,g\}$ $\{b,f\}$ $\{d,k\}$ $\{h,i\}$ $\{c,e,j,l\}$, which are

[3] Integer signals only arise at the first stage when they represent degrees of adjacent nodes.

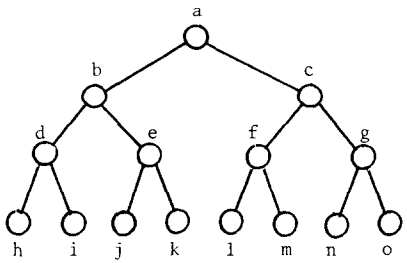

Fig. 3.

discerned by the 3rd order signatures. The successive decomposition of the node set produced by the algorithm is:

Order	Decomposition of Fig. 2
1	{a,g} {b,f} {c,d,e,h,i,j,k,l}
2	{a,g} {b,f} {d,k} {c,e,h,i,j,l}
3	{a,g} {b,f} {d,k} {h,i} {c,e,j,l}

No change thereafter

The graphs in Figs. 3 and 4 provide useful tests. The decompositions for Fig. 3 are as follows:

Order	Decomposition of Fig. 3
1	{a} {b,c,d,e,f,g} {h,i,j,k,l,m,n,o}
2	{a} {b,c} {d,e,f,g} {h,i,j,k,l,m,n,o}

No change thereafter

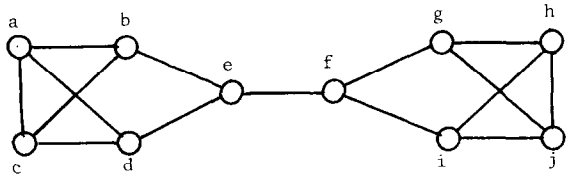

Fig. 4.

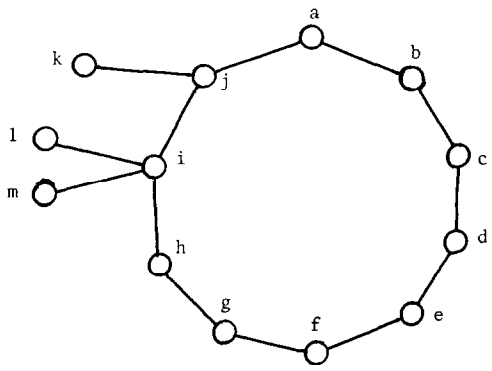

Fig. 5.

and for Fig. 4:

Order	Decomposition of Fig. 4
1	{a,b,c,d,e,f,g,h,i,j}
2	{a,c,h,j} {b,d,g,i} {e,f}

No change thereafter

So, in both cases, the automorphic equivalence classes are correctly identified by depth 2, and no higher level signatures produce any further decomposition.

The graph in Fig. 5 takes a little longer. The only two nodes that are equivalent are *l* and *m*, all of the others being role-unique. However it takes several iterations for information about the irregularities at nodes *i* and *j* to travel around the cycle. So nodes *d* and *e* are only separated by their 5th order signatures.

Order	Decomposition of Fig. 5
1	{a,b,c,d,e,f,g,h} {i} {j} {k,l,m}
2	{a} {b,c,d,e,f,g} {h} {i} {j} {k} {l,m}
3	{a} {b} {c,d,e,f} {g} {h} {i} {j} {k} {l,m}
4	{a} {b} {c} {d,e} {f} {g} {h} {i} {j} {k} {l,m}
5	{a} {b} {c} {d} {e} {f} {g} {h} {i} {j} {k} {l,m}

No change thereafter

Fig. 6.

Similarly a straight-line graph, illustrated in Fig. 6, will take roughly $n/2$ iterations for the decomposition into equivalence classes to be complete. And it is hard to imagine a graph where there could be any gain by proceeding past the $(n/2)$th order signatures.

It is interesting to note, from the above decompositions, that once a depth m is reached, such that the decomposition of the node set is the same at depth m as at $m+1$, then no greater depths produce any further decomposition. In other words, once progress in refining the partition stops, it appears to stop for good.

In fact this will always be true except in the case of networks containing cycles of length exceeding m. In the absence of long cycles, therefore, the algorithm can be halted as soon as two successive partitions are the same.

3.7. Application to directed graphs

The extension to directed graphs is straightforward but requires a little care. Everett and Borgatti (1988) very briefly describe how their algorithm (dependent on comparison of "degree vectors" from successively higher order neighborhoods) can be extended to deal with directed graphs.

They define "in-degree" and "out-degree" neighborhoods of a subset S of the node set V as follows:

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

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

Thus they defined successively higher order in-neighborhoods and out-neighborhoods for any node j. The nth order *out*-neighborhood would comprise those nodes reachable by directed-paths of length $\leqslant n$ emanating from node j. Everett and Borgatti then define a variety of vectors based upon the properties of these neighborhoods as the basis for attempting to discriminate between automorphic equivalence classes.

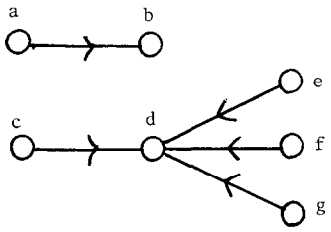

Fig. 7.

The directed graph in Fig. 7 illustrates a problem with this approach. The 1st order out-neighborhoods of *a* and *c* are {a,b} and {c,d}, respectively. All higher order out-neighborhoods are similarly {a,b} and {c,d}, because there are no more outward facing edges taking the exploration to any other part of the graph. The "out-degree vectors" for *a* and *c* are therefore both {0,1} as *a* has outdegree 1 and *b* outdegree 0. Likewise for *c* and *d*. Hence the outdegree vectors of *a* and *c* are identical at every level.

Moreover the indegree vectors of *a* and *c* are identical at every level, because each has no inward facing edges. Hence examination of in- and out-degree vectors (or any other properties of in- and out-neighborhoods as defined above) fails to separate node *a* from node *c*.

The problem with this approach is it fails to capture any "out-information" from any nodes connected only by inward facing edges. And, as in the case of Fig. 7, it similarly fails to capture any "in-information" from nodes connected only by outward facing edges.

To distinguish node *a* from node *c* information about the important structural differences between *b* and *d* has to travel *the wrong way* along the edges (a,b) and (c,d) back to *a* and *c*. Allowing the passage of information in both directions along directed links ensures that ultimately information about every part of a connected structure will reach every other part. The information flow cannot be blocked by combinations of one-way links.

With this in mind we can specify the adaptations required to extend numerical signatures to directed graphs. They are twofold.

First, at time $t(0)$, nodes should not simply transmit their in-degree or their out-degree, nor their "total" degree. They should transmit a

signal which uniquely specifies both their in-degree and out-degree, such as

$$S_{i,0} = \pi_1^{(\text{in-degree})} + \pi_2^{(\text{out-degree})} \tag{4}$$

where π_1 and π_2 should be different, and can be randomly generated.

Second, successive transmissions should occur both the right way along directed edges, and the wrong way too. Further, incoming signals should be compounded in a way that distinguishes the treatment of signals which arrived along "in-links" from those which arrived by travelling the wrong way along "out-links". One suitable compounding formula for the nth order signature at node i could be:

$$S_{i,n} = (\pi_1 + SPAN) \prod^{j} (\pi_2 + S_{j,n-1}) \prod^{k} (\pi_3 + S_{k,n-1}) \tag{5}$$

where j ranges over those nodes connected to node i by incoming links, and k runs through those connected by outgoing links. Signals received from adjacent nodes connected with symmetric links would thus be treated twice in this compounding process, distinguishing them from others.

Figure 8 is a slight adaptation of Fig. 2, with one previously symmetric link becoming unidirectional. The adaptation is sufficient to render each node of the network role unique. And the successive numerical signatures generated using expressions (4) and (5) successfully decompose the node set into automorphism classes thus:

Order	Decomposition of Fig. 8
1	{a,g} {b,f} {c} {d} {e,h,i,j,k,l}
2	{a,g} {b} {c} {d} {e} {f} {h,i,j,l} {k}
3	{a} {b} {c} {d} {e} {f} {g} {h} {i} {j} {k} {l}

No change thereafter

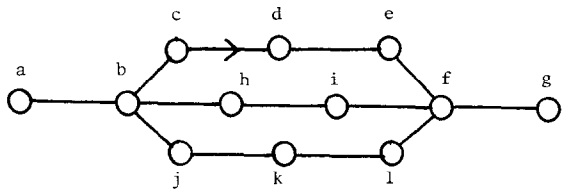

Fig. 8.

3.8. Application to multiplex graphs

It is worth observing that there are two distinct automorphism groups for multiplex networks (networks with a variety of link types). There is the group of automorphisms which regard the link types themselves as fixed; and there is the group of automorphisms which permit permutation of the link types as well as the nodes. The former will be a subgroup of the latter, and usually the two groups will be the same size.

The latter (larger) group is of theoretical interest only. We are concerned here with the former.

The technique of propagating signatures works perfectly well for multiplex graphs, using an extended version of expressions (4) and (5). The original outgoing signal from each node should reflect its degree in each link type. Simply extend expression (4), using a randomly generated set $\{\pi_i: i = 1,2,\ldots,m\}$ where m is the number of different link types.

Organized crime databases usually contain data which is both multiplex and directed, with link types such as "...is a parent of...", "...has control over...", "...receives payments from...", as well as symmetric links such as "...was seen with..." and "...works with...". To deal with such networks extend expression (4) a little further, using $2m$ different random numbers — one for each direction in each link type.

Expression (5) is similarly extensible by using either $(m + 1)$ or $(2m + 1)$ different compounding factors, depending on whether the graph is symmetric or directed.

In dealing with multiplex graphs the same principles apply as apply for directed graphs: signals should be propagated along every link at every stage, whether it be "with the flow" or against it.

This technique has one delightful property: adding to the dimensionality of the problem (by introducing multiplex ties) does not add to the dimensionality of the computations. At each successive stage signals are merely propagated from each node along every available link, and compounded according to a simple formula at the other end. The computational demands therefore remain linear with respect to the number of network links for any particular depth.

When applied practically to large networks it is most likely that one would be searching for particular structures of limited diameter,

rather than for automorphic equivalences that took account of every piece of information available in each connected subcomponent. This technique provides an extremely rapid method for finding structures identical to any specified depth, even in directed multiplex graphs. If automorphic equivalence classes really are the goal, then run the algorithm out to depth $(n/2)$ or thereabouts.

3.9. Sparse matrix techniques

The technique described is ideally suited to implementation using sparse matrix techniques (see, for instance, Hummon and Doreian (1990)). Transmission of signals at each stage is only along existing links, not "through space". So the information storage requirements at the nth depth are as follows:

(1) Maintain one original copy of the adjacency matrix, unchanged.

(2) Update a vector of numerical signatures (real numbers), one for each node.

(3) Update for each node j, a list of nodes reachable from j by paths of length not exceeding n. For application to large networks the adjacency array should be presented in sparse matrix form. Then the memory requirements for the entire process consist of two sparse binary matrices of size n (for a network with n nodes), one real-valued vector of length n, plus incidentals.

The computational burden is similarly decreased through never having to pay attention to the vast numbers of zeroes in most large adjacency arrays. Implementation of the technique for extremely large networks is therefore quite straightforward.

Use of sparse matrix techniques is even more appropriate when multiple link types exist, as large three dimensional adjacency matrices tend to consist almost entirely of zeroes. Link lists, for such networks, would identify the nodes connected and the type of the connection. This is precisely the form of many existing intelligence databases.

4. Counterexamples

It is important to understand the limitations of any algorithm. There is no guarantee that decomposition through the use of numerical signa-

tures as described will ultimately yield the automorphic equivalence classes. However, it appears that the technique does produce the automorphic equivalence classes for every type of graph it is likely to face in any practical application.

The algorithm is defeated by the AVLF graph, specially designed by four Russian mathematicians (Adelson-Velskii et al. 1969; Everett and Borgatti 1990). The AVLF graph has 26 nodes each of degree 10, and the 2nd order neighborhood of any vertex is the whole graph. The numerical signatures approach places all vertices in the same set, failing to distinguish the two disjoint automorphic equivalence classes.

This raises a dilemma. Should we complicate the algorithm in order to be able to handle the AVLF graph? Or should we preserve the usefulness of the tool by keeping it simple?

Everett and Borgatti (1988) take the former option. They noted the failure of their "degree vectors" approach to decompose the AVLF graph. They also observed that betweenness centrality did distinguish the two orbits of that graph. (Enumerating the cycles within first order neighborhoods would also have distinguished them.) So Everett and Borgatti incorporated betweenness centrality into their algorithm. They did not, however, indicate the range of graphs for which this refinement would be useful.

In a subsequent paper (Everett and Borgatti 1990) they compare the performance of different structural analysis algorithms on the AVLF graph. They observe their own is the only one known which produces the natural partition. But that is because they incorporated betweenness centrality measures into their algorithm, adding a certain degree of computational complexity.

Such an approach is perfectly reasonable if the goal is to produce an algorithm which defeats as many well known counterexamples as possible. But it may not be a prudent basis for the design of useful tools for practical use on larger networks, where computational simplicity and efficiency are at a premium.

In order to preserve the usefulness of the numerical signatures approach it should *not* be complicated any further. Its primary appeal lies in its intuitive simplicity and extraordinary efficiency. These properties should not be compromised for the sake of what could only be an unrealistic attempt to attain mathematical perfection.

4.1. Types of errors

It is useful to summarize the types of failure that could be exhibited by the numerical signatures technique in trying to determine role (automorphic) equivalence classes.

(a) Type I: Placing automorphically equivalent nodes in different classes.

(b) Type II: Placing automorphically different nodes in the same class as a result of numerical fluke.

(c) Type III: Placing automorphically different nodes in the same class as a result of generation of identical polynomials from distinct structures.

(d) Type IV: Placing automorphically different nodes in the same class as a result of deficiencies of the approach.

Type I errors should never occur unless, for some extraordinary reason, the computer is inconsistent in conducting arithmetic operations.

Type II errors are most unlikely, given the number of significant digits used in most computing environments for floating point arithmetic. They can be eliminated by increasing precision, or by repeating the process using different randomly generated compounding factors. [4]

Type III errors can be avoided entirely by using different compounding factors at each successive stage. In practice such errors do not arise even when using the same factors repeatedly.

Type IV errors seem to be rare. Except in the case of specially constructed graphs (such as the AVLF graph) the technique invariably yields the appropriate decomposition.

5. Conclusion

The generation of successive numerical signatures according to the above technique represents a simple and highly efficient method for

[4] But beware the exponential growth in numerical signatures that will occur in large networks. If real numbers are allowed to grow too fast they will hit some upper allowable limit specific to the programming language and compiler. In that case many such numbers can be assigned the same (maximum permissible) value. Avoid this by discarding the integer part of every signature at every stage. That keeps all signatures in the range (0, 1).

determining role equivalence in large networks, symmetric or directed, and with single or multiple link types.

As such it contributes substantially towards the aims laid out in the introduction to this paper. Subsequent work should develop appropriate definitions for role similarity, in different contexts, and then refine the numerical signatures technique to help uncover specific types of similarity.

In essence, the numerical signatures generated by this process at present are meaningless. Once generated they are compared with one another and then discarded. They form the basis for partitioning the node set, but the actual values themselves have no relevance. Numerical "closeness" means nothing.

Further work should adjust the means of generating and combining signals in such a way that structural similarity, appropriately defined, will result in interpretable numerical closeness.

References

Adelson-Velskii, G.M., B. Ju. Veisfeiler, A.A. Leman and I.A. Faradzev
1969 "Example of a graph without a transitive automorphism Group." *Soviet Maths Dokl 10*: 440–441.

Biggs, Norman
1974 *Algebraic Graph Theory*. London: Cambridge University Press.

Borgatti, Steve
1988 "A comment on doreian's regular equivalence in symmetric structures." University of California, Irvine. *Social Networks 10*: 265–272.

Borwein, J.M and P.B. Borwein
1987 *Pi and the AGM*. New York: Wiley and Sons Inc.

Breiger, R.L, S.A. Boorman and P. Arabie
1975 "Algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional-scaling." Harvard University; University of Pennsylvania; University of Minnesota. *Journal of Mathematical Psychology 12*: 328–383.

Burt, R.S.
1988 "Some properties of structural equivalence measures derived from sociometric choice data." *Social Networks 10*: 1–28.

Burt, R.S.
1990 "Detecting role equivalence." Columbia University. *Social Networks 12*: 83–97.

Doreian, Patrick
1987 "Measuring regular equivalence in symmetric structures." *Social Networks 9*: 89–107.

Doreian, Patrick
1988 "Borgatti toppings on doreian splits: reflections on regular equivalence." University of Pittsburgh. *Social Networks 10*: 273–287.

Everett, Martin G.
1982 "A graph theoretic blocking procedure for social networks." *Social Networks 4*: 147–168.

Everett, Martin G. and Steve Borgatti
1988 "Calculating role similarities: An algorithm that helps determine the orbits of a graph." *Social Networks 10*: 77–91.
Everett, Martin G. and Steve Borgatti
1990 "A testing example for positional analysis techniques." *Social Networks 12*: 253–260.
Faust, Katherine
1988 "Comparison of methods for positional analysis: structural and general equivalences." *Social Networks 10*: 313–341.
Heil, G.H. and H.C. White
1976 "An algorithm for finding simultaneous homomorphic correspondences between graphs and their image graphs." *Behavioural Science 21*: 26–45.
Holland, P.W., K.B. Laskey, and S. Leinhardt
1983 "Stochastic blockmodels: First steps." *Social Networks 5*: 109–138.
Hummon, N.P. and P. Doreian
1990 "Computational methods for social network analysis." *Social Networks 12*: 273–288.
Lindemann, F.
1882 "Über die Zahl π." *Math. Ann. 20*: 213–225.
Lorrain, F.P. and H.C. White
1971 "Structural Equivalence of Individuals in Networks." *Journal of Mathematical Sociology 1*: 49–80.
Panning, W.H.
1982 "Fitting Blockmodels to Data." *Social Networks 4*: 81–101.
Sailer, L.D.
1978 "Structural equivalence: meaning and definition, computation and application." University of California, Irvine. *Social Networks 1*: 73–90.
Wasserman, S. and C. Anderson
1987 "Stochastic a posteriori blockmodels: construction and assessment." *Social Networks 9*: 1–36.
Wilson, R.J.
1972 *Introduction to Graph Theory*. London: Longman Group Ltd.