Final Year Project Report
School of Computer Science
The University of Manchester

A Statistics-Based Algorithm for Positional Analysis of Networks

William Acton

Degree Programme: B.Sc. Artificial Intelligence (w/ Industrial Experience)
Supervisor: Jonathan Shapiro
Date: May 2010

Abstract

...

Project Title: A Statistics-Based Algorithm for Positional Analysis of Networks
Author: William Acton
Date: 5th May 2010
Supervisor: Jonathan Shapiro

Acknowledgements

I would like to thank my supervisor, Jon Shapiro, for his assistance, ideas, and insight.

I would also like to thank Victoria, Katie, Natalie, Karl, Mark, and my family for their patience and support.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Problem Background

Locating elements of a graph that appear to behave similarly – or *actors* that occupy the same *position* or *role* – is a problem that has been of interest to many for a long time now, especially – but not limited to – those in the field of social sciences. Graph theory is often used as an intuitive model, where the vertices of a graph represent individuals and the edges between the vertices represent particular relationships between the individuals. With just this simple model, a complex web of ties can develop very quickly. Analysing the structure of this resulting network can give a great deal of insight into the properties and the behaviour of the graph elements – much more than mere visual inspection can.

Structural analysis can assign metrics to the vertices of the graph, measures that can be used to help compare and contrast the elements and to form judgements about them. Such a judgement is whether or not two elements are similar, or even equivalent. The knowledge of the presence or absence, possibly even the extent, of structural similarity between individuals is an important asset in social studies.

One significant difficulty associated with identifying elements that are equivalent is that there are a few different perceptions of equivalence (detailed later in section 2.4), varying with strictness. For example, the strictest notion of equivalence states that two elements are equivalent if they are connected by the same relations to exactly the same elements (Burt, Positions in Networks, 1976). In contrast, a more general equivalence states that two elements are equivalent if they share the same *types* of neighbours (White & Reitz, 1983).

Knowing which level of equivalence to use is clearly dependent upon the task, but is not always obvious. As highlighted by Borgatti and Everett (1992), researchers have been known to misidentify the most rigid notion of structural equivalence – identical connections – with the more abstract idea of social roles; this is exemplified by the following claim:

"Even more important, if a researcher uses structural equivalence as a criterion, he can identify positions, or sets of individuals, that correspond to social roles in the network of communication." (Caldeira, 1988, p. 46)

This misidentification of equivalences leads researchers to use a different notion of equivalence than the one they specified. For example, Galaskiewicz and Krohn wrote,

"A key concept in our study is the social role. . . . Two actors are in the same relative position in social space, i.e., have the same role, to the extent that they have similar relationships to others in the social arena or fields. . . . To phrase this in the context of resource dependency theory, two actors occupy the same structural position, i.e., role in the network, to the extent that they are dependent upon the same organizations for the procurement of needed input resources and the same actors are dependent upon them for their output resources." (pp. 528-529)

Galaskiewicz and Krohn proceeded to use a more rigid, absolute equivalence to identify organisational roles (p. 532) instead of the relative equality they described.

A further difficulty is the choice of measures used in the analysis of the network. There are a number of graph-theoretic properties that may be considered, such as degree, betweenness, closeness, reach, etc. (The most commonly used metrics are described in section 2.3) Certainly a combination of these is required to produce good estimates of similarity, and all of them must be taken into account to achieve the greatest accuracy. Considering many properties becomes unwieldy and infeasible, so in reality many of these measures are omitted in favour of the most influential ones. But in what circumstances should, for example, betweenness measures be used in favour of closeness measures, and vice versa? This is a hard question for which there is no clear-cut answer.

1.2. Project Proposal

The proposal was to devise a statistics-based algorithm that would cluster together the similar elements of a graph. In doing so, it was hoped that the strict mathematical boundaries of graph-theoretical equivalence would not have to be adhered to; the algorithm would be capable of grouping together elements that, although they will ideally be highly similar, are not necessarily equivalent by any of the formal definitions.

This flexibility is hoped to provide an additional viewpoint when clustering elements of graphs. As an analogy – one that will be used throughout this report – consider the slightly contrived case of mothers. Imagine there are three mothers, the first with a single child, the second with eight children, and the third with ten. One definition of equivalence says these mothers are all different as they have a different number of children. The definition of equivalence one generalisation above this says that all the mothers are the same as they all have children. There is a thought that it would be useful if we could distinguish the first mother from the other two – those with *many* children – as this is a more realistic and intuitive grouping. This is what the project aims to fulfil.

1.3. Existing Work

There exists extensive work into defining the different concepts of equivalence – and thus the different notions of roles, or positions, of nodes – in relation to the elements of graphs. Following from this, algorithms on how to cluster together elements of graphs based on a chosen equivalence definition have been published.

There are three major, well-known definitions of equivalence that have been well-researched and documented: *structural equivalence*, by Lorrain and White (1971), Burt (1976), and Breiger, Boorman, and Arabie (1975); *automorphic equivalence* or *structural isomorphism*, by Everett (1985) and Winship (1988); and *regular equivalence*, by White and Reitz (1983) and Borgatti and Everett (1989). These are explained fully in section 2.4. There are also other general or abstract equivalences looked at by Winship and Mandel (1983), Breiger and Pattison (1986), and Hummell and Sodeur (1987), but these are out of the scope of what is needed for the project.

Everett and Borgatti have presented notable algorithms to help cluster elements of a graph based on automorphic equivalence (Calculating Role Similarities: An Algorithm that Helps Determine the Orbits of a Graph, 1988) and on regular equivalence (Two Algorithms for Computing Regular Equivalence, 1993). The essential points of the original algorithm for computing the extent of

regular equivalence, REGE, were presented by White in three unpublished papers (1980) (1982) (1984) (Borgatti & Everett, 1993).

To the author's knowledge, there has been no prior work on statistical clustering of graph elements based upon the aforementioned perceptions of equivalence.

1.4. Objectives

The primary objective was to develop an algorithm that clusters the elements of a graph based on statistical measures. The resulting clustering was to resemble a middle-way between two generalisation levels of equivalence, providing a third, alternative clustering, as described above. This can be broken down into three milestones:

Milestone 1: Implement an existing algorithm to be used as a reference point, which the new algorithm will be compared against. This algorithm will apply a relatively strict equivalence (automorphic equivalence) to constrain the size of the clusters. Therefore a more flexible approach to clustering the elements of the same graph will be expected to have larger-sized clusters.

Milestone 2: Develop and implement a new algorithm, primarily based on statistical measures such as the mean. The clustering that is produced should resemble that produced by the existing algorithm; elements that are grouped together via the rigid graph-theoretic approach should still be grouped together in this new approach.

Milestone 3: Evaluate the clustering produced by the new algorithm against that produced by the existing algorithm. There should be a numerical measure that can easily communicate the similarity between the two clusterings. Ideally this will show that the clusterings are neither identical nor dissimilar, and that clustered-together elements in the existing algorithm are clustered together in the new one.

1.5. Report Overview

Chapter 2 will elucidate, in detail, the different equivalence concepts before explaining the background mathematics and graph theory required to understand what Everett and Borgatti's existing algorithm does and why. The algorithm itself will then be explained.

Chapter 3 will present the proposed new statistics-based algorithm and explain the techniques used, before describing the testing that took place and the similarity measures developed in order to evaluate the output from the two algorithms.

Chapter 4 will detail the reasoning behind the implementation and technical choices made, and will demonstrate some of the more interesting and difficult aspects of the development.

Chapter 5 will consider the result of the new algorithm and will show how to use the algorithm in the environment it was developed.

Chapter 6 concludes the report, with reflections on goal achievement and on the further activity that could improve the algorithm, and the overall tool, to enhance its usefulness.

2. Background

2.1. Notions of Position

Position refers to how an actor[1] interacts with the rest of the network, and therefore has connotations of structural correspondence or similarity. Actors who are connected in the same way to the rest of the network are said to occupy the same position. This in turn leads to the notion of equivalence; those actors that occupy the same position – have the same role – are equivalent (Borgatti & Everett, 1992).

2.1.1. Actor Similarity vs. Actor Cohesion

The phrase "*connected in the same way to the rest of the network*" used above can, however, be interpreted in at least two fundamental ways.

The first is in a literal sense, and is the most basic interpretation. The underlying clustering principle in this case is cohesion, or proximity; two actors are connected in the same way to the rest of the network if they are tied to exactly the same nodes. The second interpretation is a metaphorical one. Here the underlying principle is similarity; two actors are connected in the same way to the rest of the network if the actors they are linked to also exhibit similarity.

Borgatti and Everett (1992) adapt a problem presented by Hofstadter (1985) to illustrate this distinction and ask the reader to consider the following two abstract structures:

$$A = \langle\, 1\,2\,3\,4\,5\,5\,4\,3\,2\,1 \,\rangle \text{ and } B = \langle\, 0\,1\,2\,3\,4\,4\,3\,2\,1\,0 \,\rangle$$

"Hofstader asks, "What is to B as 4 is to A? Or, to use the language of roles: What plays the role in B that 4 plays in A?" (p. 549). *According to Hofstader, an overly literal, concrete answer is 4, whereas a more natural, more analogical response is 3."* (p. 3)

This distinction is also exemplified in mathematics. In the case of geometry, two rectangles are the same if corresponding sides are of equal length – this is a literal interpretation of "the same". Under the metaphorical interpretation, two rectangles would be declared the same if they are similar; if they have the same aspect ratio (and therefore the same proportions).

2.1.2. Positional Analysis

Positional analysis, therefore, aims to partition actors into mutually exclusive sets of equivalent actors who share *similar* interactivity with the network. In other words, the analysis groups together actors that play the same *role* in the network. The contrasting cohesive approach (Burt, 1978)

[1] Note that the following terms are used interchangeably throughout this report. Their use is dependent on the context of the material and the level of abstraction employed.

- Graph, network;
- Node, vertex, actor, point, graph element;
- Edge, relationship, link, tie, connection;
- Position, role, type.
- Position, role, type.

(Friedkin, 1984) aims to identify closely related actors – those that have a high proximity and share identical neighbours.

2.2. Technical Definitions and Notation

In this section, the basic technical definitions and notation used throughout this report are given. These definitions will be used in subsequent sections to build upon and define more complex ideas, so it is important that they are clarified early on. The definitions and notations used here correspond with those given by Borgatti and Everett (1992).

Graphs or networks are represented as graphs denoted $G(V, E)$, where V refers to a set of vertices (or nodes, actors, points, graph elements) and where E refers to a set of edges (or relationships, links, ties, connections). The vertex sets of graph G s and H are represented as $V(G)$ and $V(H)$ respectively. Similarly, the edge sets of the two graphs are represented as $E(G)$ and $E(H)$.

Furthermore, the position of node a is denoted $P(a)$. The position of a node is an attribute of that node that can be, and is, used to classify the node – that is, the node is identified as being of a specified type, occupying a particular position, or playing a certain role. Such categorisation allows positions to be instinctively visualised as colours (Everett & Borgatti, 1990). Extending positional notation, let $P(\{a, b, \dots\}) = \{P(a) \cup P(b) \cup \dots\}$, i.e., if S is a set of nodes then $P(S)$ is the set of distinct positions occupied by the nodes in set S.

In an undirected graph, the notation $N(a)$ is defined as the set of nodes directly connected to node a, so that $N(a) = \{c: (a, c) \in E\} = \{c: (c, a) \in E\}$. $N(a)$ is referred to as the neighbourhood of a. If S is a subset of V then $N(S) = S \cup \{v \in V - S: (v, s) \in E, s \in S\}$; nodes that are linked directly to the subset S are part of their neighbourhood.

This project does not take into account directed graphs, but for the purposes of completeness their neighbourhoods are defined as follows. The set of nodes that a receives ties from is known as the in-neighbourhood, $N^{in}(a)$, is defined as $N^{in}(a) = \{c: (c, a) \in E\}$. Likewise, the set of nodes that a sends ties to is known as the out-neighbourhood, $N^{out}(a)$, is defined as $N^{out}(a) = \{c: (a, c) \in E\}$. The neighbourhood $N(a)$ of node a then is defined as the ordered pair $N(a) = \left(N^{in}(a), N^{out}(a)\right)$.

From these simple definitions it can be seen that more interesting expressions can be built. For example, $P\big(N(a)\big)$ – the set of positions occupied by the nodes with ties to a – and $N\big(P(a)\big)$ – the set of nodes that are linked to a node by the position that is occupied by a.

To take a simple example and to demonstrate the above concepts, consider the basic graph shown in Figure 1 below:

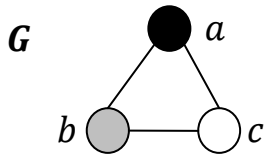

Figure 1: Graph with 3 vertices and 3 edges.

V or, more precisely, $V(G)$, is the vertex set $\{a, b, c\}$, and E or $E(G)$ is the edge set $\{ab, ac, bc\}$. Here the nodes have been coloured to enable uncomplicated representation of positions. Therefore, $P(a)$ is black $P(b)$, is grey, and $P(c)$ is white. Additionally $P(G)$, is the set $\{black, grey, white\}$. This graph is undirected so $N(c)$ is simply the set of nodes $\{a, b\}$. To move to the slightly more complicated examples, $P\big(N(b)\big)$ is the set of positions $\{black, white\}$ and $N\big(P(b)\big)$ is the set of nodes that are neighbours to nodes of a $grey$ position, i.e., $\{a, c\}$.

Nodes have attributes; if an attribute makes no reference to the names or labels of the nodes in a graph then these are classified as *structural* or *graph-theoretic attributes*. For example, the property of being directly connected to every other node in the graph is a structural attribute. The property of being directly connected to node b is not a structural attribute. In the example in Figure 1 above, a exhibits both of these properties but only the former is a structural attribute.

2.3. Graph Theory

This section builds upon the mathematics of the previous section and introduces some graph-theoretic concepts. These will be required to understand the foremost equivalence definitions as well as the algorithms currently used to implement the notion of position as equivalence.

2.3.1. Mapping: Isomorphism

Two graphs, G and H, are isomorphic if there is a one-to-one mapping from the nodes of G to the nodes of H that preserves the adjacency of nodes (Wasserman & Faust, 1994). Thus an isomorphic graph preserves the relationships among the objects of the original graph once a one-to-one mapping has been undertaken. Therefore, if two nodes are connected in the original graph, their corresponding nodes in the second, isomorphic graph will also be connected. A formal definition for this is that a graph isomorphism between two graphs G and H is a mapping $\pi: G \to H$ such that for all $a, b \in V(G), (a, b) \in E(G) \leftrightarrow \big(\pi(a), \pi(b)\big) \in E(H)$ (Borgatti & Everett, 1992).

An example of two isomorphic graphs and the mapping between them is given in Figure 2 and Table 1 below.

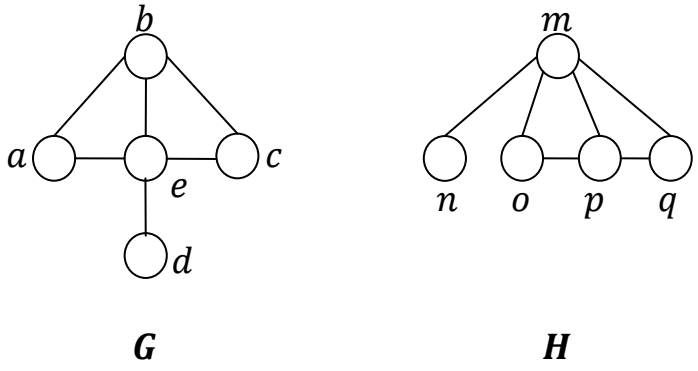

Figure 2: Graphs G and H are isomorphic.

Table 1: The graph isomorphism $\pi_1 : G \rightarrow H$ relating the graphs in Figure 2.

g	$\pi_1(g)$
a	o
b	p
c	q
d	r
e	s

Isomorphic graphs are, unsurprisingly, identical with respect to all graph-theoretic properties. A useful way of thinking about whether or not two graphs are isomorphic is to imagine morphing one on top of the other. If, once the two graphs were stripped of their vertex and edge labels, one of the graphs could be picked up off the paper and then rearranged and scaled so that it can be placed perfectly on top of the other graph, they are isomorphic. At this point, without labels, the two graphs would be identical. From this it is clear that the graphs would have identical graph-theoretic properties. The only possible differences between isomorphic graphs are non-structural attributes – any labels of the nodes and edges.

2.3.2. Mapping: Automorphism

All graphs are isomorphic with themselves. More formally, for all graphs there exists a mapping $\pi: G \rightarrow G$ such that π is an isomorphism. This isomorphic mapping of a structure to itself is known as an *automorphism*. π can always be the identity mapping (for all $v \in V, \pi(v) = v$) – which is why all graphs have at least one automorphic mapping – but graphs often have automorphic mappings that are not the identity.

As an example taken from Borgatti and Everett (1992), consider the graph in Figure 3. By observing the symmetries of the graph, one can easily perceive its automorphisms. If all labels were removed, a rotation of 180° along the horizontal axis would prove indistinguishable from the original graph. Likewise, a rotation of 180° along the vertical axis would have a similar effect, as would a composition of the two rotations. Table 2 shows all the automorphism of the graph.

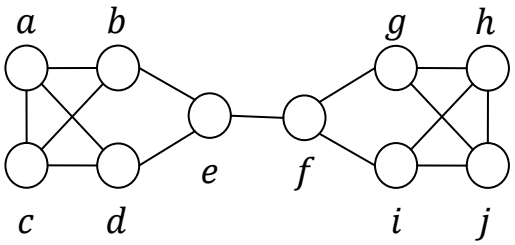

Figure 3: Graph with four automorphisms.

Table 2: All automorphisms of the graph in Figure 3.

v	$\pi_1(g)$	$\pi_2(g)$	$\pi_3(g)$	$\pi_4(g)$
a	h	c	j	a
b	g	d	i	b
c	j	a	h	c
d	i	b	g	d
e	f	e	f	e
f	e	f	e	f
g	b	i	d	g
h	a	j	c	h
i	d	g	b	i
j	c	h	a	j

Upon examination of Table 2 one notices that groups of nodes become interchangeable. For example, when any node from the set $\{a, c, h, j\}$ is mapped automorphically, the mapping can only ever be to a node from that same set. These sets formed by isomorphic actors are known as *orbits*. The orbits of the graph in Figure 3 are $\{a, c, h, j\}$, $\{b, d, g, i\}$, and $\{e, f\}$.

2.3.3. Centrality: Degree

The three significant centralities were first comprehensively defined by Freeman (1978). The first of these is *degree centrality*. Degree centrality is a straightforward index of the activity of a node, a count of the number of adjacencies it has:

$$C_D(v_j) = \sum_{i=1}^{n} f(v_i, v_j) \text{ where } f(v_i, v_j) = \begin{cases} 1 \text{ if and only if } v_i \text{ and } v_j \text{ are linked} \\ 0 \text{ otherwise} \end{cases}$$

$C_D(v_j)$ will be large if v_j has many direct neighbours and will be small if v_j has little direct contact. Consequently, $C_D(v_j) = 0$ if v_j is in total isolation and is not connected to any other node in the network.

To further utilise the notation used by Borgatti and Everett, the *degree vector* of a graph G is defined by $C_D(G) = (C_D(v_1), C_D(v_2), \dots, C_D(v_n))$ for all $v_i \in V(G)$ where $n = |V|$, $C_D(v_i) \leq C_D(v_{i+1})$, for $i = 1, \dots, n-1$.

Take graph G in Figure 4 below as an example.

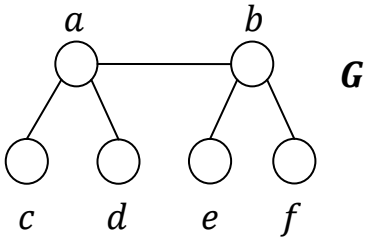

Figure 4: Graph with 6 nodes, 5 edges, and 2 orbits.

$C_D(c) = 1$, as does $C_D(d)$, whereas $C_D(a) = 3$. The degree vector $C_D(G) = (1, 1, 1, 1, 3, 3)$. Moreover, if S is a subset of V then $\langle S \rangle$ is the induced subgraph, i.e., the graph that would remain if only the vertices in set S existed. $C_D(S)$ will be written in place of $C_D(\langle S \rangle)$ for the purposes of readability. Hence, continuing with the above example, if the vertex subset $S = \{a, b, c, d\}$ then $C_D(S) = (1, 1, 1, 3)$. The subgraph $\langle S \rangle$ is depicted in Figure 5 below.

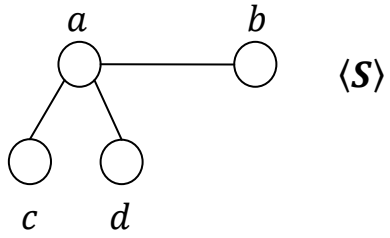

Figure 5: Subgraph of graph in Figure 4.

In order to compare the centralities of nodes from different graphs, this measure of degree can be normalised over $n - 1$, as a given node can be adjacent to at most $n - 1$ other nodes in a graph. Doing this removes the effect of network size, allowing relative comparison.

2.3.4. Centrality: Betweenness

The second view of centrality, as described again by Freeman (1978), is based upon the frequency with which a node lies on the shortest (geodesic) paths between pairs of other points. This is known as *betweenness centrality*. Consider the graph in Figure 5. There are six geodesics, as shown in Figure 6 below:

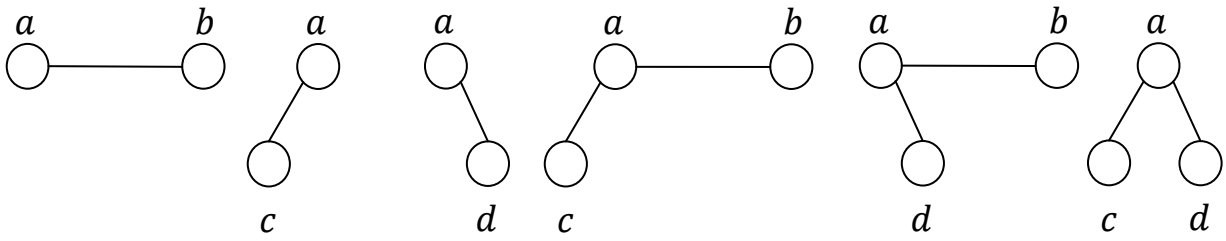

Figure 6: The six geodesics of the graph in Figure 5.

Here it can be seen that a lies on all of the shortest paths between every node in the graph; the first three geodesics are of length one and represent the shortest path from a to every other node. The final three geodesics show the shortest paths between the remaining pairs of nodes bc, bd, and cd. This means that a must be navigated in order for any two nodes to communicate, that a exhibits a potential for control of their communication. It is this potential for control that defines betweenness. Thus, the greater number of geodesics a node lies upon, the greater its betweenness.

Note that there may be more than a single geodesic between a pair of nodes. This must be taken into consideration when calculating betweenness. Let g_{ij} be the number of geodesics linking n_i and n_j, and $g_{ij}(v_k)$ be the number of those geodesics that contain v_k. $\frac{g_{ij}(v_k)}{g_{ij}}$ then gives the probability that node v_k falls on a randomly selected geodesic connecting v_i with v_j.

To calculate the overall betweenness centrality of node v_k, the probabilities of v_k lying on a randomly selected shortest path between two nodes must be summed for all pairs of nodes. However, node-pairs that are directly connected (and therefore need not traverse through any additional nodes) are ignored.

$$C_B(v_k) = \sum_{i}^{n} \sum_{j}^{n} \frac{g_{ij}(v_k)}{g_{ij}} \text{ where } i < j, i \neq j \neq k$$

In a similar manner to the degree vector, the *betweenness vector* of a graph G is defined by $C_B(G) = \big(C_B(v_1), C_B(v_2), \dots, C_B(v_n)\big)$ for all $v_i \in V(G)$ where $n = |V|$, $C_D(v_i) \leq C_D(v_{i+1})$, for $i = 1, \dots, n-1$.

Again consider the subgraph in Figure 5. As a lies on three geodesics that do not include a as either the start or end node, $C_B(a) = 3$. By the same token, the remaining nodes all have a betweenness of 0 as they are never on the shortest path between two other nodes. To demonstrate a slightly more complex example, take graph G shown in Figure 7 below and the matrix of shortest paths in Table 3:

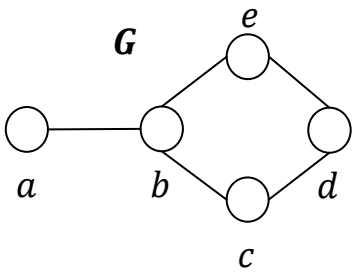

Figure 7: Graph where some node-pairs have two geodesics.

Table 3: Matrix of shortest paths of graph in Figure 7.

	a	**b**	**c**	**d**	**e**
a		-	b	b, c b, e	b
b			-	c e	-
c				-	b d
d					-
e					

As a is not seen within the matrix, $C_B(a) = 0$. That is, none of the shortest paths found in the graph contain a. b, however, can be seen numerous times. Its betweenness is calculated by summing the probabilities of b occurring in the shortest path, i.e.:

Node b lies on the single geodesic between a and c, therefore its probability of occurrence is 1. Likewise, b lies on both of geodesics between a and d and on the single path between a and e. Of the two shortest paths between c and e, b can only be found on one of them. Therefore, the probability in this case is 0.5. Consequently, $C_B(b) = 1 \div 1 \div 1 \div 0.5 = 3.5$.

The rest of the nodes have the following betweenness values: $C_B(c) = 1$, $C_B(d) = 0.5$, $C_B(e) = 1$. The betweenness vector of graph G is $C_B(G) = (0, 0.5, 1, 1, 3.5)$.

As with degree centrality, betweenness centrality is dependent on the size of the network. For some applications it again may be useful to have a relative measure to compare betweenness of nodes from separate graphs. Freeman (1978) proved that relative betweenness can be expressed as the ratio $\frac{2C_B(v_k)}{n^2-3n\div2}$.

2.4. Equivalence Definitions

This section explores the three foremost varying notions of position, as explained by Borgatti and Everett (1991) (1992). Subsequently, the equivalence selected as the template to aim towards in the new algorithm is justified.

2.4.1. Position as Structural Equivalence

Structural equivalence is the most strict, or least general, of the three equivalences discussed at here. Under this notion an actor's position is determined solely by its local neighbourhood: if G is graph and $a, b \in V$, then $P(a) = P(b)$ if and only if $N(a) = N(b)$[2]. In other words, an actor's position is defined by those actors it has direct connections to; a and b are structurally equivalent if a and b have identical relationships.

To draw on the "mothers" analogy used previously in the report, this is akin to saying that two mothers are equivalent if they are both mothers of the same child. Obviously this is impossible because of the nature of the "is-a-mother-of" relation, but serves as a comprehensible parallel.

Looking at graph G in Figure 4 it can be seen that nodes c and d are structurally equivalent because they both have a single relationship with node a. Similarly, e and f are structurally equivalent because of their connection with node b. However, a and b are not structurally equivalent because

[2] This definition, although sufficient for the purposes of this report, has an important shortcoming: in graphs without reflexive loops, nodes with a direct connection to each other cannot occupy the same position. A slight alteration to the definition that resolves this problem is $P(a) = P(b)$ if and only if $N(a) - \{a, b\} = N(b) - \{a, b\}$. However, this version too is not without problem. There is fault when a single directed arc links a and b, and when trying to distinguish nodes with reflexive loops from ones without (Borgatti & Everett, 1992).

The following definition provided by Everett, Boyd, and Borgatti (1990) works in all cases: $P(a) = P(b)$ if and only if $\exists \pi \in Aut(G)$ such that $\pi = (a\ b)$ and $\pi(a) = b$, where $Aut(G)$ denotes the automorphism group of a graph G.

a has ties to c and d while b has ties to e and f. This can be visualised as a *role colouring*, as demonstrated in Figure 8 below; actors of the same colour play the same role shortcoming

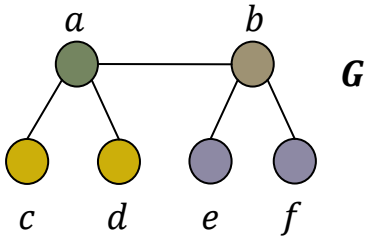

Figure 8: Graph in Figure 4 coloured according to structural equivalence.

Locating structurally equivalent actors produces cohesive subsets, as only direct adjacencies are taken into account. This reveals as much about the proximity of actors as it does their similarity. Of course, two actors with identical neighbourhoods will have a high degree of similarity. On the other hand, structural equivalence is incapable of detecting similarity *in spite of* proximity, a trait that has clear advantages.

2.4.2. Position as Automorphic Equivalence

Automorphic equivalence – also often used interchangeably with the term *structural isomorphism*, but in the interests of clarity only referred to as automorphic equivalence in the rest of this report – is an equivalence that observes similarities between two nodes' interactivity within a network regardless of the identity of the nodes in their neighbourhoods.

This is accomplished by viewing two actors as occupying the same position, not if they are tied to the same actors, but if the actors they are tied to correspond. Whereas the neighbours of structurally equivalent actors contain the same actors, the neighbours of automorphically equivalent actors contain the same *positions*. Formally, if actors a and b are isomorphic, then for all $c \in V, (a, c) \in E$ implies there exists an actor d isomorphic to c such that $(b, d) \in E$ (Borgatti & Everett, 1992).

Under this approach, $P(a) = P(b)$ if and only if $P\big(N(a)\big) = P\big(N(a)\big)$; if the positions occupied by the neighbourhood of a is equal to the positions occupied by the neighbourhood of b, a and b are automorphically equivalent. Actors that occupy the same positions in this automorphic sense can be interchanged in a mapping, and therefore belong to the same orbit (as described earlier in section 2.3.2). Thus, finding automorphically equivalent actors becomes a case of calculating the orbits of a graph.

Table 4: All automorphisms of the graph in Figure 4.

$\boldsymbol{v}$	$\boldsymbol{\pi_1(g)}$	$\boldsymbol{\pi_2(g)}$	$\boldsymbol{\pi_3(g)}$	$\boldsymbol{\pi_4(g)}$
$\boldsymbol{a}$	a	a	b	b
$\boldsymbol{b}$	b	b	a	a
$\boldsymbol{c}$	d	c	e	f
$\boldsymbol{d}$	c	d	f	e
$\boldsymbol{e}$	e	f	c	d
$\boldsymbol{f}$	f	e	d	c

Table 4 above shows all the automorphisms of the graph in Figure 4. Using this, one can identify the orbits of the graph as $\{a, b\}$ and $\{c, d, e, f\}$. Two orbits correspond to two different types of nodes (positions) in the graph. Once again this can be visualised as a role colouring as depicted in Figure 9 .

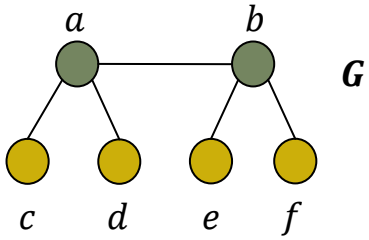

Figure 9: Graph in Figure 4 coloured according to automorphic equivalence.

Continuing with the "mothers" analogy, automorphic equivalence would say that two mothers are the same if they have the same number of children.

2.4.3. Position as Regular Equivalence

Regular equivalence is the least strict, or most general, of the three equivalences discussed, and an abstraction above automorphic equivalence. Returning to the "mothers" analogy once again, here two mothers are the same if they have children, regardless of how many.

Whereas automorphic equivalence enumerates how many of each type one particular type is connected to, regular equivalence is satisfied as long as there exists at least one connection to all the necessary types. As an example, if graph G in Figure 4 was to be extended by connecting a new node, g, to b and coloured in accordance to regular equivalence, one possible colouring can be seen in Figure 10 below:

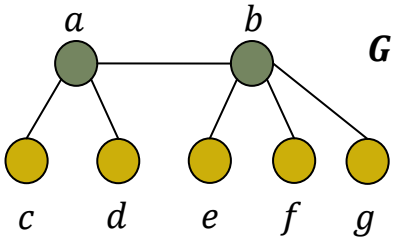

Figure 10: Extension of graph in Figure 4 coloured according to regular equivalence.

Here each green node is linked to at least one other green node and one yellow node. Each yellow node is linked to one green node. However, automorphic equivalence is now broken because if one green node has ties to two yellow nodes, a green node elsewhere cannot have ties to three yellow nodes.

Herein lies the significant drawback of regular equivalence: its leniency. Although there are undoubtedly applications where disregarding the quantity of a type of neighbour is useful, it seems

counter-intuitive. For most practical purposes it inherently feels astute to distinguish between actors that have a single connection to a given type of actor from actors that have hundreds of such connections.

2.4.4. Selected Equivalence & Justification

From the three equivalences discussed in this chapter, automorphic equivalence seems to be the most obvious target for the new algorithm. Not that the new algorithm will simulate automorphic equivalence, but it will aspire to resemble a somewhat more relaxed version of it. Clearly too much leniency will result in regular equivalence, so there must also be a level of restraint.

As mentioned earlier, the weakness of regular equivalence is its ability to judge two actors as equivalent independent of how many type x nodes may be connected to it. On the other end of the spectrum is structural equivalence, which is far too strict for most realistic intentions. It would be unusual, for example, to declare that two surgeons play the same role only if they operate on the same patients. A more reasonable assertion is that two surgeons play the same role if they operate on the same types of patients; surgeons that operate on patients with brain tumours are neurosurgeons. This is an illustration of regular equivalence, and works well in this instance.

Automorphic equivalence is almost a good halfway point between the two extremes. In the "mothers" analogy, two mothers are regularly equivalent if they have children, even if one mother has one child and the other has ten. The two mothers are automorphically equivalent if they have the same number of children. On the face of it this appears to be a sensible to claim, but there is a pragmatism behind grouping mothers of nine children together with mothers of ten because in reality there is not going to be much discernable difference. In other words, there is benefit in assigning mothers to intermediary groups: those with few children, those several children, and those with a lot of children.

This is something that automorphic equivalence is too strict to allow, whereas regular equivalence is too lenient. It is perceivable, however, that taking a statistics-based approach to this problem could result in a clustering of actors that is indicative of this intermediate view.

The new approach needed something to compare its clustering to. It was conceived that the results were going to be resemble the clustering produced under the view of automorphic equivalence more so than the extremely general view of regular equivalence. There would also be many more clearly defined sets of actors that would be useful in the analysis and evaluation of the new approach. This explains why the notion of position as automorphic equivalence was chosen to be the chief objective to aim towards and to be compared against.

2.5. Existing Algorithm

Everett and Borgatti (1988) outline an algorithm that helps determine the orbits of a graph. This in turn partitions the vertices of a graph into distinct automorphically equivalence classes. Their algorithm is presented in this section.

2.5.1. Additional Mathematical Definitions

The basic technical definitions in section 2.2 are expanded upon here, as they are required to understand the algorithm proposed by Everett and Borgatti.

Higher-order neighbourhoods are defined inductively: $N^{i\div1}(S) = N\left(N^i(S)\right)$ where S is a subset of a vertex set V. To reinforce this, consider the graph in Figure 4. If $S = \{c\}$ then the immediate neighbourhood $N(S) = \{a, c\}$, the second-order neighbourhood (neighbours of neighbours) $N^2(S) = N\big(N(S)\big) = N(\{a, c\}) = \{a, b, c, d\}$, and the third-order neighbourhood $N^2(S) = \{a, b, c, d, e, f\}$ as c is no more than three links from every other node in the graph.

If $x \in V$ then the *point-deleted neighbourhood* $\tilde{N}^i(x)$ is defined as $N^i(x) - \{x\}$. In the current example where $S = \{c\}$ then $N(S) = \{a, c\}$ and $N^2(S) = \{a, b, c, d\}$ whereas $\tilde{N}(S) = \{a\}$ and $\tilde{N}^2(S) = \{a, b, d\}$. The subgraphs produced by these two point-deleted neighbourhoods are shown in Figure 11 below. From looking at these we can easily observe their degree vectors. $C_D\left(\tilde{N}(S)\right) = (0)$ and $C_D\left(\tilde{N}^2(S)\right) = (1, 1, 2)$.

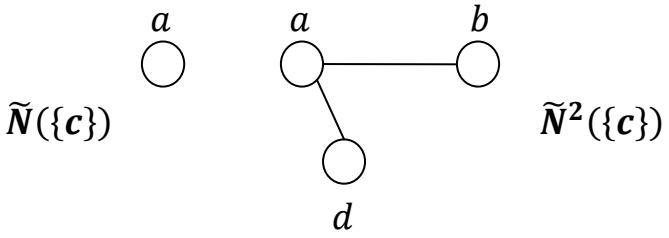

Figure 11: Point-deleted neighbourhoods of graph in Figure 4.

2.5.2. Finding Orbits

The algorithm proposed by Everett and Borgatti is based on the fact that any property not shared by two vertices guarantees that they are in separate orbits. To illustrate this, consider Figure 9, in which each orbit has been assigned a distinct colour. Partitioning the vertex set of G by degree presents two sets, $\{a, b, c, d\}$ that have a degree of 1 and $\{e, f\}$ that have a degree of 3. This has found the orbits of the graph. Usually, however, the problem will not be so simple. For more complex graphs there is a need to examine the degrees of the neighbours of a node, as opposed to the node itself. Everett and Borgatti (1988) provide the following demonstration of this using the graph in Figure 12 below:

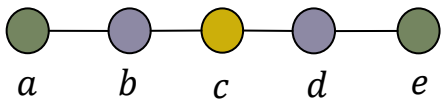

Figure 12: Graph with 5 nodes and 4 edges, coloured to highlight its orbits.

The orbits of this graph are $\{a, e\}$, $\{b, d\}$, and $\{c\}$. Partitioning the vertex set with respect to degree only, the obtained sets are $\{a, e\}$ and $\{b, c, d\}$. If the neighbourhoods, and higher-order

neighbourhoods, of the nodes are inspected instead, the following Table 5 is obtained:

Table 5: Degree vectors of neighbourhoods of the graph in Figure 12.

x	$\widetilde{N}(x)$	$C_D\left(\widetilde{N}(x)\right)$	$\widetilde{N}^2(S)$	$C_D\left(\widetilde{N}^2(x)\right)$
a	$\{b\}$	(0)	$\{b,c\}$	$(1,1)$
b	$\{a,c\}$	$(0,0)$	$\{a,c,d\}$	$(0,1,1)$
c	$\{b,d\}$	$(0,0)$	$\{a,b,d,e\}$	$(1,1,1,1)$
d	$\{c,e\}$	$(0,0)$	$\{b,c,e\}$	$(0,1,1)$
e	$\{d\}$	(0)	$\{c,d\}$	$(1,1)$

Examining the degree vectors of immediate (point-deleted) neighbourhoods distinguishes a and e from b, c, and d. Further examination of the degree vectors of second-order neighbourhoods then distinguishes c from b and d. Continuing to look at yet higher-order neighbourhoods produces more accurate results. In this instance, a second-order neighbourhood was all that was required to identify the orbits. For larger graphs this may not be the case.

It is also important to note that the actual orbits of the graph can only be finer partitions than those found at the highest-order neighbourhood considered. Therefore, a result that contained partitions $\{a\}$, $\{b,c\}$, and $\{d\}$, for example, has correctly distinguished those three sets as being different. The actual orbits, however, may be $\{a\}$, $\{b\}$, $\{c\}$, and $\{d\}$. The partition of b and c will occur if the neighbourhood order is high enough, but at the cost of computation time.

Formally, x and y belong to the same set if and only if $C_D\left(\widetilde{N}^i(x)\right) = C_D\left(\widetilde{N}^i(y)\right)$ for $i \geq 0$.

Although difficult, it is possible to find examples on which this method fails. A refinement of the above technique can be made to combat this; the inclusion of betweenness vectors as well as degree vectors. The combination of these two concepts, then, supplies the partitioning criteria: two vertices, $x, y \in V$ belong to the same set if the neighbourhoods are similar in respect to both degree *and* betweenness. That is, provided:

(i) $C_D\left(\widetilde{N}^i(x)\right) = C_D\left(\widetilde{N}^i(y)\right), i = 1 \ldots n-1$

(ii) $C_B\left(\widetilde{N}^i(x)\right) = C_B\left(\widetilde{N}^i(y)\right), i = 1 \ldots n-1.$

2.5.3. Measures of the Extent of Equivalence

The above method only identifies equivalence. Everett and Borgatti (1988) proceed to devise a simple measure of the extent of equivalence. First they define a function $SIM(i,x,y)$ that compares the degree and betweenness of actors x and y at neighbourhood level i, returning 1 if they are identical. Technically,

$$SIM(i,x,y) = \begin{cases} 1 \; if \; C_D\left(\widetilde{N}^i(x)\right) = C_D\left(\widetilde{N}^i(y)\right) \\ and \; C_B\left(\widetilde{N}^i(x)\right) = C_B\left(\widetilde{N}^i(y)\right) \\ 0 \; otherwise. \end{cases}$$

Further refinements can be made to return the proportion of identical elements in the vectors if they are not identical.

Everett and Borgatti then define the extent of equivalence between two actors x and y as $A(x,y)$, the proportion of identical neighbourhoods:

$$A(x,y) = \frac{\sum_{i=1}^{n-1} SIM(i,x,y)/i}{\sum_{i=1}^{n-1} 1/i}$$

Therefore, actors that differ in their immediate neighbourhoods will be assigned a value close to 0, if not, 0. Whereas actors that differ in higher-order neighbourhoods closer to $n-1$ are not penalised as heavily, and will score a value close to 1. Those that are identical in all neighbour levels will be assigned a perfect similarity of 1.0.

2.5.4. Pseudocode

The pseudocode for this algorithm, adapted from Everett and Borgatti, can be written as follows:

- Initialise equivalence matrix; $A(x,y) = 0$ for all x, y
- For each neighbourhood level $i = 1 \ldots n-1$ do:
  - For each actor $x = 1 \ldots n$ do:
    - Generate neighbourhood $\widetilde{N}^i(x)$
    - Compute degree vector $C_D\left(\widetilde{N}^i(x)\right)$
    - Compute betweenness vector $C_B\left(\widetilde{N}^i(x)\right)$
  - For each pair of actors $x = 2 \ldots n$ and $y = 1 \ldots x-1$ do:
    - $A(x,y) = A(x,y) \div SIM(i,x,y)/i$
    - $A(x,y) = A(x,y)/\sum\left(\frac{1}{i}\right)$

Applying this algorithm (considering up to third-order neighbourhoods) to the graph in Figure 12 produces the following equivalence matrix $A(x,y)$:

Table 6: Everett and Borgatti (1988) algorithm applied to graph in Figure 12.

	a	***b***	***c***	***d***	***e***
a	1.000	0.591	0.455	0.591	1.000
b	0.591	1.000	0.761	1.000	0.591
c	0.455	0.761	1.000	0.761	0.455
d	0.591	1.000	0.761	1.000	0.591
e	1.000	0.591	0.455	0.591	1.000

Table 6 above shows which actors are equivalent – those with a value in the matrix of 1.000 – along with the similarity of actors that are not equivalent. For example, no actors are equivalent to c, but the orbit $\{b,d\}$ is more similar to c than the orbit $\{a,e\}$. This was also suggested by the difference in the vectors and their magnitudes in Table 5. Here a value is given to that idea.

2.5.5. Complexity

The worst case complexity of this algorithm occurs when the maximum order of the neighbourhood encompasses every other node, for every node in the graph. Therefore, approximately n degrees and betweennesses have to be calculated per node, or n times. This means the algorithm has a quadratic worst case time complexity, n^2, or, in big O notation, $O(n^2)$.

3. New Algorithm

The proposed statistics-based algorithm utilises *k-means clustering*. Given k clusters, this analysis will partition the elements into the clusters with the closest mean.

3.1. k-Means Clustering Description

The following definition of k-means clustering was adapted from lecture notes written by Shapiro (2009) although first published by Lloyd in 1982 (Least Sqaures Quantization in PCM).

Clusters in the sense have three key factors: means, subsets, and the number of clusters. The mean of the data in cluster i is a vector of dimension D, and is denoted $m(i)$. S_i represents the set of data in cluster i. The subset S_i consists of all data points closest to the $i$the mean than any other mean. k refers to the number of clusters, and is selected in advance.

The objective of k-means clustering is to find the clustering that minimises the *quantisation error*,

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x_n \in S_i} \|x_n - m(i)\|^2.$$

The standard algorithm to perform this clustering is implemented as an iterative refinement technique. It consists of an initialisation step proceeded by an assignment step and an update step. The final two steps are repeated until convergence, at which point there will be no change in the cluster assignments.

Initialising the means of the clusters is often performed based on a chosen heuristic, though it may be random. The following two steps are then repeated until the algorithm stabilises:

1. **Associate** the data with the cluster that has the closest mean:

$$S_i = \{x \mid x \in D, \|x - m(i)\|^2 \leq \|x - m(j)\|^2 \; for \; all \; j\} \; for \; all \; i$$

2. **Update** the means of the clusters based on the data now associated with them:

$$m(i) = \frac{1}{|S_i|} \sum_{x \in S_i} x; for \; all \; i$$

3.2. Cluster Means

The proposed algorithm performs two stages of k-means clustering. The first stage clusters the elements of the graph according to degree. This preliminary clustering forms the foundation of the overall clustering. After all, two nodes of the same degree are more similar (from the viewpoint of automorphic equivalence) than two nodes of differing degree.

Consequently, if the range of degrees is greater than k, some nodes of differing degrees will be grouped together. The clusters are initialised with an even distribution to ensure that only nodes of similar degrees are grouped together, i.e., that a cluster that contains nodes with a degree of three

and four do not contain nodes of degree six when there exists a cluster containing nodes with a degree of five.

With this grouping of nodes with a similar activity, a secondary clustering then takes place. This secondary clustering focuses on neighbour *types*. An assumption is made that the number of sets produced by the initial clustering is synonymous with how many positions there are in the network. Each cluster now represents a certain position.

The secondary clustering calculates the mean number of connections each cluster has to every other cluster. In other words, how many ties to a neighbouring type, on average, each node in a group should have. This is how the second stage is initialised. The clustering then begins again. This time an assignment is made by looking at the amount of typed neighbours each node *actually* has.

This is where verbal explanations begin to get complicated, so consider the graph in Figure 9 for a graphical example. Let us assume the initial clustering on degree has produced the two sets depicted by the colouring. That is, two sets have been identified: cluster 1 contains $\{c, d, e, f\}$ with an average degree of 1, and cluster 2 $\{a, b\}$ with an average degree of 3. Repeating the assignment process according to degree will not change the clustering because the average degree is now the actual degree.

Initialisation of the secondary clustering would have the effect shown in the coloured Table 7:

Table 7: Results of clustering based on neighbour type of graph in Figure 9.

	Type 1 Neighbours	Type 2 Neighbours
Cluster 1	0	1
Cluster 2	2	1

On average, elements of the first cluster have no ties to elements of that same cluster, whereas they have a single tie with elements of the second cluster. In a similar fashion, elements of the first cluster have two type 1 neighbours and a single link to a type 2 node.

When assigning the data points to one of these clusters, a direct comparison cannot be done because there are an arbitrary number of dimensions to consider. As such, the *mean squared error*, $\sum_{i=1}^{n}\left(\overline{x_i} - \overline{x'_i}\right)^2$, is used.

Ideally, this neighbour-position based clustering will bring together elements that may have been initially narrowly separated by their degree but were closely related by the types of nodes they are connected to; analogous to grouping together two mothers, one with nine children and one with ten, after having originally been distinguished as dissimilar due to the differing number of children.

3.3. Pseudocode

Bearing in mind the assignment and update equations given in section 3.1, the pseudocode for this particular k-means clustering algorithm is, where $\{x_1, \ldots, x_n\}$ are the nodes in the network:

- Initialise degree means of k clusters, for $i = 1 \dots k$ do:
  - $m(i) = \left(C_{D_{max}}/k\right) \times \left((2i) \div 1\right)$
- While degree means of the clusters have not stabilised:
  - For $i = 1 \dots k$ do (assign elements to clusters):
    - $S_i = \{\}$
    - For $v = 1 \dots n$ do:
      - If $\|x_v - m(i)\|^2 \leq \|x_v - m(j)\|^2$ for all $j = 1 \dots k, i \neq j$
      - $S_i = S_i \cup \{x_v\}$
  - For $i = 1 \dots k$ do (update cluster means):
    - $m(i) = \frac{1}{|S_i|} \sum_{x \in S_i} x$
- Initialise neighbour type means of resulting clustering for $i = 1 \dots k, S_i \neq \emptyset$ do:
  - For $j = 1 \dots k$ do:
    - $m(i)_j = \sum \sum_{v=1}^{n} \widetilde{N}(x_v) \in S_j$ (count the neighbours of type S_j)
    - $m(i)_j = m(i)_j / |S_i|$
- While neighbour type means of the clusters has not stabilised:
  - Assign elements to clusters as above
  - Update cluster means as computed in the initialisation step

3.4. Complexity

When considering the time complexity of this suggested algorithm, there are a number of factors to take into account. To assign nodes to clusters, k number of n distances have to be calculated for each k. This is a quadratic time complexity of $O(k^2 n)$. During the updating of cluster means, a maximum of n nodes are accounted for per cluster, giving a linear complexity of $O(kn)$. Therefore the overall time complexity is $O(k^2 n)$.

This quadratic complexity is similar to the algorithm described in section 2.5, with one significant difference. When the size of a graph grows but the number of clusters remains the same, the automorphic role similarity algorithm has a quadratic growth whereas this algorithm has linear growth. It should also be noted that it is unlikely for the number of clusters to ever be greater than or equal to the number of nodes in the graph. That is, every actor plays a completely independent role. Realistically, the number of clusters in a large network is likely to be a small proportion of the number of nodes. Thus, for most intents and purposes, the k-means approach will be the faster of two algorithms.

3.5. Testing

3.5.1. K-Means Clustering Stabilisation

3.5.2. Clustering Tests

3.5.3. Measures of Similarity Tests

4. Implementation

4.1. GML?

4.2. Why Java? JGraphT library?

4.3. Important & Difficult Aspects

5. Results

5.1. Required Interaction

5.2. Interpretation of Output

6. Conclusions

6.1. Achievement of Goals

6.2. Changes to Original Plan

6.3. Further Activity

7. References

Borgatti, S. P., & Everett, M. G. (1992). Notions of Position in Social Network Analysis. *Sociological Methodology , 22*, 1-35.

Borgatti, S. P., & Everett, M. G. (1989). The Class of All Regular Equivalences: Algebraic Structure and Computation. *Social Networks , 11*, 65-88.

Borgatti, S. P., & Everett, M. G. (1993). Two Algorithms for Computing Regular Equivalence. *Social Networks* , 361-376.

Breiger, R. L., & Pattison, P. E. (1986). Social Networks. *Cumulated Social Roles: The Duality of Persons and their Algebras , 8*, 215-256.

Breiger, R. L., Boorman, S., & Arabie, P. (1975). An Algorithm for Clustering Relational Data with Applications to Social Network Analysis. *Journal of Mathematical Psychology , 12*, 329-383.

Burt, R. S. (1978). Cohesion Versus Structural Equivalence as a Basis for Network Subgroups. *Sociological Methods and Research , 7*, 189-212.

Burt, R. S. (1976). Positions in Networks. *Social Forces , 55*, 93-122.

Caldeira, G. A. (1988). Legal Precedent: Structures of Communication Between State Supreme Courts. *Social Networks , 10*, 29-55.

Everett, M. G. (1985). Role Similarity and Complexity in Social Networks. *Social Networks , 7*, 353-359.

Everett, M. G., & Borgatti, S. P. (1990). Any Colour You Like It. *Connections , 13*, 19-22.

Everett, M. G., & Borgatti, S. P. (1988). Calculating Role Similarities: An Algorithm that Helps Determine the Orbits of a Graph. *Social Networks , 10*, 77-91.

Everett, M. G., & Borgatti, S. P. (1991). Role Colouring a Graph. *Mathematical Social Sciences , 21*, 183-188.

Everett, M. G., Boyd, J. P., & Borgatti, S. P. (1990). Ego-Centered and Local Roles: A Graph-Theoretic Approach. *Journal of Mathematical Sociology , 15*, 163-72.

Freeman, L. C. (1978). Centrality in Social Networks: Conceptual Clarification. *Social Networks , 1*, 215-239.

Friedkin, N. E. (1984). Structural Cohesion and Equivalence Explanations of Social Homogeneity. *Sociological Methods and Research , 12*, 235-261.

Galaskiewicz, J., & Krohn, K. R. (1984). Positions, Roles, and Dependencies in a Community Integrated System. *Sociological Quarterly , 25*, 527-550.

Hofstadter, D. R. (1985). *Metamagical Themas: Questing for the Essence of Mind and Pattern.* New York: Basic Books.

Hummell, H., & Sodeur, W. (1987). Strukturbeschreibung von Positionen in sozialen Beziehungsnetzen. *Methoden der Netzwerkanalyze* , 1-21.

Lloyd, S. P. (1982). Least Sqaures Quantization in PCM. *IEEE Transactions on Information Theory , 28*, 129-137.

Lorrain, F. P., & White, H. C. (1971). Structural Equivalence of Individuals in Networks. *Journal of Mathematical Sociology , 1*, 49-80.

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *Introduction to Information Retrieval.* Cambridge University Press.

Shapiro, J. L. (2009). Lecture 4.1 - Unsupervised Learning I: Competitive Learning. *CS6483* .

Wasserman, S., & Faust, K. (1994). *Social Network Analysis: Methods and Applications.* New York: Cambridge University Press.

White, D. R. (1982). Measures of global equivalence in social networks. *Unpublished manuscript* .

White, D. R. (1984). REGGE: a REGular Graph Equivalence algorithm for computing role distances prior to blockmodeling. *Unpublished manuscript* .

White, D. R. (1980). Structural equivalences in social networks: concepts and measurement of role structures. *Unpublished manuscript* .

White, D. R., & Reitz, K. (1983). Graph and semigroup homomorphisms on semigroups of relations. *Social Networks , 5*, 193-234.

Winship, C. (1988). Thoughts about Roles and Relations: An Old Document Revisited. *Social Networks , 10*, 209-231.

Winship, C., & Mandel, M. J. (1983). Roles and Positions: A Critique and Extension of the Blockmodeling Approach. *Sociological Methodology, 1983-84* , 314-344.

8. Appendices