A note on matricial ways to compute Burt’s structural holes in networks*

Alessio Muscillo

Department of Economics and Statistics, Università di Siena, piazza San Francesco 7, 53100, Italy

June 8, 2021

Abstract

In this note I derive simple formulas based on the adjacency matrix of a network to compute measures associated with Ronald S. Burt’s structural holes (effective size, redundancy, local constraint and constraint). This can help to interpret these measures and also to define naive algorithms for their computation based on matrix operations.

Keywords: network measures, structural holes, effective size, redundancy, constraint, computation

1 Introduction

In the last decades, the social context in which economic activities are embedded has become more and more the focus of attention and research (Granovetter, 1985; Schweitzer et al., 2009; Goyal, 2018). Regularities of network structures that shape – and, in turn, are shaped by – economic behavior have been studied with the increased awareness that “designing many economic policies requires a deep understanding of social structure” (Jackson et al., 2017).

To analyze how the network structure relates to economic behavior, different notions and measures are used to capture an individual’s importance, influence, or centrality in a network (Bloch et al., 2019). One of the most fascinating concept is that of structural holes, developed by Burt (2009), which refers to the absence of connections between groups and to the fact individuals might

---

*email: alessio.muscillo2@unisi.it – ORCID iD: 0000-0002-2648-4272

The author acknowledges funding from the Italian Ministry of Education Progetti di Rilevante Interesse Nazionale (PRIN) grant 2017ELHNNJ.

The author thanks Paolo Pin, Tiziano Razzolini, Claudia Ruzza and Gabriele Lombardi for their help and support. Additional material and Python code used in this note are available online here. 
benefit from establishing links that fill voids and bridge gaps. The versatility of this concept has stimulated the definition of several measures, each capturing different aspects in different applications (Burt, 2004; Goyal and Vega-Redondo, 2007; Rubí-Barceló, 2017). However, this has also generated confusion when it comes to which exact measure has to be computed, how to compute it and what are the relations with similar measures (Borgatti, 1997; Newman, 2018; Everett and Borgatti, 2020). Moreover, calculating these measures directly by applying the definition formulas can be very slow and computationally intensive, because it would require looping over each node’s neighbors (and its neighbors’ neighbors).

In this note, I consider the main measures associated with structural holes, namely effective size, redundancy, local constraint and constraint, and derive simple formulas from the adjacency matrix of the network. This might help to interpret these measures and also produces intuitive and naive algorithms based on matrix multiplications which work fairly well on moderately large networks (see Figure 2). However, while this approach is clean and simple, it has clear limitations when the network under analysis is very large. In such a case, one should avoid storing explicitly the matrices and should preferably rely on distributed algorithms and more advanced techniques for triangle listing with vertex orderings and neighborhood markers (Chiba and Nishizeki, 1985; Li et al., 2019).

2 Notation

In what follows, matrices are denoted by capital letters (e.g. $A$, $P$) and their elements denoted by the corresponding letter with subscripts (e.g. $a_{ij}$, $p_{ij}$). Generic nodes of a network (i.e., a graph) will be indicated by $i$, $j$ or $k$. Consequently, an adjacency matrix will be indicated by $A = (a_{ij})_{i,j=1,...,n}$, where $n$ is the number of nodes and the elements $a_{ij}$ can be 0 or 1 for binary networks or generic real numbers for weighted networks.

Vectors and their elements will be respectively denoted by bold letters (e.g. $\mathbf{x}$, $\mathbf{y}$) and letters with a (single) subscript (e.g. $x_i$). The vector obtained by taking the diagonal elements of a square matrix $A$ is denoted by $\text{Diag}(A)$ and, analogously, the matrix that has $\mathbf{x}$ as its diagonal and 0s elsewhere is denoted by $\text{Diag}(\mathbf{x})$. The transposed of a vector or matrix is denoted by $\cdot^T$ (e.g. $\mathbf{x}^T$, $A^T$). Hereafter, vectors are considered as columns, that is $(n \times 1)$-matrices and their transposed as row vectors $\mathbf{x}^T$. Accordingly, the (matrix) multiplication of a column vector $\mathbf{x}$ times a row vector $\mathbf{y}^T$ will give a matrix (e.g. $\mathbf{x}\mathbf{y}^T \in \mathbb{R}^{n \times n}$) whereas $\mathbf{x}^T\mathbf{y}$ a scalar.

The matrix multiplication between two matrices $A$ and $B$ will be denoted by juxtaposition, i.e. $AB$, whereas element-by-element operations such as element-wise multiplication or division will be denoted respectively by $\odot$ and $\oslash$. The $n$-dimensional unitary vectors in $\mathbb{R}^n$ containing all 0s but one 1 in $i$-th position is denoted by $\mathbf{e}_i$, while the vector containing all 1s is denoted by $\mathbf{1}$. The identity matrix is denoted by $I$. 

2
3 Effective size and redundancy for undirected binary networks

The original definition of effective size and redundancy in Burt’s works was complicated, but Borgatti (1997) has shown that it can be simplified. Here, we consider an undirected and binary network with no self-loops. The intuitive idea (see Figure 1) is first to compute a node’s redundancy, which is the mean number of connections from a neighbor to other neighbors. Then, the effective size is obtained by subtracting the redundancy to the node’s degree.

Let \( r_i \) be the redundancy of node \( i \) and let \( t_i \) be “the number of ties in the network (not including ties to ego)” (Borgatti, 1997).\(^1\) Then, the redundancy is simply

\[
\bar{r}_i = \frac{2t_i}{d_i},
\]

where \( d_i \) is \( i \)'s degree. Notice that \( r_i \) goes from 0 to \( d_i - 1 \).\(^3\) Then, the effective size \( s_i \) of node \( i \) is defined as:

\[
s_i = d_i - r_i.
\]

Now, let us see how to compute this in a matricial form. Let \( A = (a_{ij})_{i,j} \) be the adjacency matrix of such an undirected and binary network and let \( d = (d_i)_i \) be the vector of nodes’ degrees.\(^4\) Notice that for a binary network, the elements of the square \( A^2 \) count the number of common neighbors. Indeed, for every two nodes \( i \) and \( j \), the \((i,j)\)-th element of \( A^2 \) is:

\[
(A^2)_{ij} = \sum_{k=1}^{n} a_{ik}a_{kj} = |\{ k : k \in N(i) \text{ and } k \in N(j) \}|
\]

\[
= \text{number of common neighbors of } i \text{ and } j,
\]

since \( a_{ik} \) is different from 0 if and only if \( i \) and \( k \) are linked and, analogously, \( a_{kj} \) is different from 0 if and only if \( k \) and \( j \) are linked. Obviously, we only want to count the common neighbors for pairs of nodes that are actually linked in the network. To do so, it suffices to multiply \( A^2 \) element by element for \( A \) itself. Lastly, we want to sum all these numbers and divide them by the corresponding degree.

Summing up, a matricial way to compute the vector of nodes’ effective size, \( s = (s_i)_i \), is by computing the following vector:

\[
s = d - (A^2 \odot A) \mathbf{1} \odot d,
\]

\(^{1}\)Note: “ego” here is node \( i \).

\(^{2}\)Since the network is assumed undirected, the links of \( t_i \) have to be counted twice.

\(^{3}\)It is also well related to the notion of local clustering, which can be thought of as a normalized version of redundancy ranging from 0 to 1. It can easily be shown that the relationship between the local clustering \( q_i \) and redundancy \( r_i \) of a node \( i \) of degree \( d_i \) is given by: \( q_i = \frac{r_i}{d_i - 1} \) (Newman, 2018).

\(^{4}\)\( A \) is a symmetric matrix only containing 0s and 1s. In such a case the vector of nodes’ degree can be obtained in several ways, for example as \( d = \text{Diag}(A^2) \) or \( d = A\mathbf{1} \).
Description of redundancy and effective size

Figure 1: Adapted from Burt’s and Borgatti’s works. Let us compute the effective size for node $A$. Consider one of its neighbors, say $G$. Then, $A$ and $G$ have 3 “common neighbors”: $B$, $E$ and $F$. Analogously for all 4 neighbors of $A$ (respectively, $B$, $E$, $F$ and $G$). Summing up all these numbers and divide them by $A$’s degree, which is 4, gives: $\frac{1+1+1+3}{4} = \frac{6}{4} = 1.5$. Lastly, $A$’s effective size is its degree minus its redundancy: $4 - 1.5 = 2.5$. In the example of this section, the matricial computation is also done for the remaining nodes.

where $A^2$ is $A$ squared with the standard matrix multiplication. The $i$-th component of such a vector, $s_i$, is node $i$’s effective size. By definition, the redundancy is just the last term, that is $r = (A^2 \odot A)1 \odot d$, where $r = (r_i)_i$.

Example Consider the network in Figure 1. The adjacency matrix and the degree vector are, respectively:

$$A = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}, \quad d = \begin{pmatrix}
4 \\
3 \\
1 \\
2 \\
6
\end{pmatrix}.$$

Now, since

$$A^2 = \begin{pmatrix}
4 & 1 & 1 & 2 & 1 & 1 & 3 \\
1 & 3 & 1 & 1 & 2 & 2 & 2 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 \\
2 & 1 & 1 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 2 & 2 & 1 \\
1 & 2 & 1 & 1 & 2 & 2 & 1 \\
3 & 2 & 0 & 1 & 1 & 1 & 6
\end{pmatrix},$$

\footnote{For simplicity, in $A$ the 0s are not indicated. Notice that self loops are not allowed.}
then equation (4) gives the effective size for each node:

\[
\begin{pmatrix}
A \\
B \\
C \\
D \\
E \\
F \\
G \\
\end{pmatrix}
\rightarrow
s = \begin{pmatrix}
2.5 \\
1.667 \\
1 \\
1 \\
1 \\
1 \\
4.667 \\
\end{pmatrix}
\]

nodes’ effective size.

4 Local constraint (a.k.a. dyadic constraint)

Let \( A = (a_{ij})_{i,j} \) be the adjacency matrix of a network (not necessarily binary or unweighted).\(^6\) Following Everett and Borgatti (2020), the local constraint on \( i \) with respect to \( j \), denoted \( \ell_{ij} \), is defined by\(^7\)

\[
\ell_{ij} = \left( p_{ij} + \sum_{k \in N(i) \setminus \{j\}} p_{ik}p_{kj} \right)^2,
\]

where \( N(i) \) is the set of neighbors of \( i \) and \( p_{ij} \) is the normalized mutual weight of the edges joining \( i \) and \( j \), that is,

\[
p_{ij} = \frac{a_{ij} + a_{ji}}{\sum_k (a_{ik} + a_{ki})}.
\]

Notice that, assuming absence of self-loops, every \( p_{ii} = 0 \), because \( a_{ii} = 0 \). This implies that the second term in definition (5) can be written as

\[
\sum_{k \in N(i) \setminus \{j\}} p_{ik}p_{kj} = \sum_{k \in N(i)} p_{ik}p_{kj} - p_{ij} p_{jj} = \sum_{k \in N(i)} p_{ik}p_{kj},
\]

and, hence, \( \ell_{ij} \) becomes

\[
\ell_{ij} = \left( p_{ij} + \sum_{k \in N(i)} p_{ik}p_{kj} \right)^2.
\]

Now, let us focus on \( p_{ij} \), writing equation (6) in matricial terms:\(^8\)

\[
p_{ij} = (A + A^T)_{ij} \cdot \frac{1}{((A^T + A)e_i)^T 1},
\]

---

\(^6\)That is, \( A \) is not necessarily symmetric and may contains elements different from 0 and 1. The only assumption here is that no self-loop is allowed, that is, \( a_{ii} = 0 \) for all nodes \( i \).

\(^7\)This is also known as dyadic constraint. The definition used in NetworkX’s algorithm for local constraint is slightly different. The only modification consists in changing \( k \in N(i) \setminus \{j\} \) with \( k \in N(j) \). I discuss how to adapt the matricial algorithm in the additional material available at the link in the first acknowledgements note.

\(^8\)Notice that the denominator here is the multiplication of a row vector times a column vector, which is a number.
and let us define vector $x = (x_i)_i$, where
\[ x_i = (A + AT)e^T_i = \sum_k (a_{ik} + a_{ki}), \] (10)
Thus
\[ x = (A + AT)^T1 = (AT + A)1 \] (11)
and we can consider the vector containing all inverted elements:
\[ y = 1 \odot x = \begin{pmatrix} 1/x_1 \\ \vdots \\ 1/x_n \end{pmatrix} \] (12)
Then, define the matrix which only consists on the diagonal being equal to $y$, that is, $Diag(y)$. Now, we can finally compute $P = (p_{ij})_{i,j}$ as follows:
\[ P = Diag(y) \ (A + AT). \] (13)

Now, let us focus on $\ell_{ij}$. Consider again the second term of the definition’s formula as written in equation (8)
\[ \sum_{k \in N(i)} p_{ik}p_{kj} = \sum_k a_{ik}p_{ik}p_{kj}, \] (14)
where the summation on the right-hand side is over all nodes $k$ (not just limited to $i$’s neighbors).\(^9\) Written in matricial form, this summation in equation (14) can simply be expressed as\(^{12}\)
\[ (A \odot P)P, \] (15)
where $\odot$ is the element-wise matricial multiplication and the second is a matrix multiplication.
To conclude, we can write the matrix $L = (\ell_{ij})_{i,j}$ containing all links’ local constraints as follows:
\[ L = \left[ P + (A \odot P)P \right] \odot \left[ P + (A \odot P)P \right]. \] (16)

Summing up, the algorithm takes the adjacency matrix $A$ as input and proceeds with the following steps:
1. $x = (A + AT)1$;
2. $y = 1 \odot x$;
3. $P = Diag(y) \ (A + AT)$;
4. $L = \left[ P + (A \odot P)P \right] \odot \left[ P + (A \odot P)P \right]$.

\(^9\)Notice that $AT + A$ is always symmetric, even if $A$ is not.
\(^{10}\)By pre-multiplying a diagonal matrix, we are just multiplying every row $i$ of $(A + AT)$ for the corresponding element $y_i$ of the diagonal.
\(^{11}\)If the network is weighted, then here one has to first compute the binary version $A$ of the weighted adjacency matrix $W$, where $a_{ij} = 1$ if and only if $w_{ij} \neq 0$ and $a_{ij} = 0$ otherwise. Then, one can just apply the formula written in the text.
\(^{12}\)The modification mentioned in Footnote 7 consists in taking here $P(P \odot A)$.
5 Constraint

Let $L = (\ell_{ij})_{i,j}$ be the local constraint matrix computed in equation (16). According to Everett and Borgatti (2020), the constraint for node $i$ is\(^{13}\)

$$c_i = \sum_{j \in N(i)} \ell_{ij}.$$  

One can re-write this as follows:\(^{14}\)

$$c_i = \sum_j \ell_{ij}a_{ij},$$

where $(a_{ij})_{i,j} = A$ is the adjacency matrix. So, the vector $c = (c_i)_{i}$ containing the constraints of the network is obtained by summing the rows of the matrix $L \odot A$:\(^{15}\)

$$c = [1^T (L \odot A)]^T.$$  

\(^{13}\text{Notice that in our notation } N(i) \text{ does not include } i \text{ itself. To be even more clear, one could then write } c_i = \sum_{j \in N(i) \setminus \{i\}} \ell_{ij}.\)

\(^{14}\text{In case the network is weighted, then here the matrix } A \text{ is the binary version of the weighted adjacency matrix } W, \text{ as observed in Footnote 11.}\)

\(^{15}\text{Remember that in our notation vectors are always considered as columns.}\)
Comparison with NetworkX’s routines

Figure 2: A basic comparison of computational speed between this paper’s algorithms (orange) and NetworkX’s algorithms (blue) for effective size (left) and constraint (right), as the number of nodes increases from 1,000 to 10,000. The networks are Barabasi-Albert graphs with parameter $m = 5$ (top row) and Erdos-Renyi random graphs with parameter $p = 0.01$ (bottom row). Notice, however, that a precise comparison in a controlled environment should show that more efficient methods scale better and are better suited for large networks.

References

BLOCH, F., JACKSON, M. O. and TEBALDI, P. (2019). Centrality measures in networks. Available at SSRN 2749124.

BORGATTI, S. P. (1997). Structural holes: Unpacking Burt’s redundancy measures. Connections, 20 (1), 35–38.

BURT, R. S. (2004). Structural holes and good ideas. American Journal of Sociology, 110 (2), 349–399.

— (2009). Structural holes: The social structure of competition. Harvard University Press.

CHIBA, N. and NISHIZEKI, T. (1985). Arboricity and subgraph listing algorithms. SIAM Journal on Computing, 14 (1), 210–223.
EVERETT, M. G. and BORGATTI, S. P. (2020). Unpacking Burt’s constraint measure. Social Networks, 62, 50–57.

GOYAL, S. (2018). Heterogeneity and networks. In Handbook of Computational Economics, vol. 4, Elsevier, pp. 687–712.

— and VEGA-REDONDO, F. (2007). Structural holes in social networks. Journal of Economic Theory, 137 (1), 460–492.

GRANOVETTER, M. (1985). Economic action and social structure: The problem of embeddedness. American Journal of Sociology, 91 (3), 481–510.

JACKSON, M. O., ROGERS, B. W. and ZENOU, Y. (2017). The economic consequences of social-network structure. Journal of Economic Literature, 55 (1), 49–95.

LI, F., ZOU, Z., LI, J., LI, Y. and CHEN, Y. (2019). Distributed parallel structural hole detection on big graphs. In International Conference on Database Systems for Advanced Applications, Springer, pp. 519–535.

NEWMAN, M. (2018). Networks. Oxford University Press.

RUBÍ-BARCELÓ, A. (2017). Structural holes in social networks with exogenous cliques. Games, 8 (3), 32.

SCHWEITZER, F., FAGIOLO, G., SORNETTE, D., VEGA-REDONDO, F., VESPIGNANI, A. and WHITE, D. R. (2009). Economic networks: The new challenges. Science, 325 (5939), 422–425.