Computing Cliques and Cavities in Networks

Dinghua Shi

Shanghai University

Zhifeng Chen

Fujian Normal University

Xiang Sun

Fujian Normal University

Qinghua Chen

Fujian Normal University

Chuang Ma

Anhui University

Yang Lou

City University of Hong Kong

Guanrong Chen (✉ eegchen@cityu.edu.hk)

City University of Hong Kong

Computing Cliques and Cavities in Networks

Dinghua Shi[1]*, Zhifeng Chen[2], Xiang Sun[2], Qinghua Chen[2]*, Chuang Ma[3], Yang Lou[4], Guanrong Chen[4]*

([1]Department of Mathematics, College of Science, Shanghai University, China, shidh2012@sina.com;

[2]School of Mathematics and Statistics, Fujian Normal University, China, 920978196@qq.com;

[2]School of Mathematics and Statistics, Fujian Normal University, China, sun_xiang_3@163.com;

[2]School of Mathematics and Statistics, Fujian Normal University, China, qinghuachen@fjnu.edu.cn;

[3]Department of Internet Finance, School of Internet, Anhui University, China, chuang_m@126.com;

[4]Department of Electrical Engineering, City University of Hong Kong, China, felix.lou@my.cityu.edu.hk;

[4]Department of Electrical Engineering, City University of Hong Kong, China, eegchen@cityu.edu.hk)

Abstract: Complex networks have complete subgraphs such as nodes, edges, triangles, etc., referred to as cliques of different orders. Notably, cavities consisting of higher-order cliques have been found playing an important role in brain functions. Since searching for the maximum clique in a large network is an NP-complete problem, we propose using k-core decomposition to determine the computability of a given network subject to limited computing resources. For a computable network, we design a search algorithm for finding cliques of different orders, which also provides the Euler characteristic number. Then, we compute the Betti number by using the ranks of the boundary matrices of adjacent cliques. Furthermore, we design an optimized algorithm for finding cavities of different orders. Finally, we apply the algorithm to the neuronal network of C. elegans in one dataset, and find all of its cliques and some cavities of different orders therein, providing a basis for further mathematical analysis and computation of the structure and function of the C. elegans neuronal network.

Keywords: C. elegans neuronal network, boundary matrix, clique, cavity, 0-1 programming, Euler characteristic number, Betti number

Introduction

A network has three basic sub-structures: chain, star and cycle. Chains are closely related to the concept of average distance, while a small average distance and a large clustering coefficient together implies a small-world network[1], where the clustering coefficient is determined by the number of triangles, special cycles. Stars follow heterogeneous degree distributions, with which the growth of node numbers and a preferential attachment mechanism together leads from random networks[2] to scale-free networks[3]. Cycles contain not only triangles but also higher-order cliques and cavities. The cycle structure brings redundant paths into network connectivity, creating feedback effects and higher-order interactions in network dynamics.

In retrospect, we introduced the notion of totally homogeneous networks[4] in studying optimal network synchronization, which are networks with the same node degree, same girth (length of the smallest cycle passing the node) and same path-sum (sum of all distances from other nodes to the node). We showed[4] that totally homogeneous networks are the easiest ones to self-synchronize among all networks of the same size. Recently, we found[5] that cycles are essentially described by the Euler characteristic number (alternative sum of cliques of different orders) and the Betti number (number of cavities of different orders), while higher-order cliques and smallest cavities are key components of the totally homogenous networks.

It is more challenging to study network cycles than node degrees. A triangle is the smallest first-order cycle (denoted 1-cycle for brevity), which consists of three edges, and is a second-order clique (denoted 2-clique for brevity). Similarly, a complete graph of four nodes, which consists of 4 triangles, is a 3-clique. In the same manner, these concepts can be extended to higher-order ones. In a connected undirected network, the number of cycles with different lengths (defined as the number of cliques that compose the cycle) is huge, therefore new mathematical concepts and tools are needed[6,7], including such as cyclic operations and equivalent cycles, to classify them and select their representatives for effective analysis and computation.

In the studies of brain science, computational neuroscience has a special focus on cyclic structures in neuronal networks. It was found, for example as reported in [8], that cycles generate neural loops in the brain, which not only can transmit information all over the brain but also have an important feedback function. It was suggested[8] that this provides a foundation for the brain functions of memories and controls. Unlike cliques, which are placed at some particular locations e.g. cerebral cortexes, cavities extend to almost everywhere in the brain connecting many different regions together. In [9], it points out that in both biological and artificial neural networks, one can find huge numbers of cliques and cavities therein, which are massive and complex but not noticed before. Of particular importance is that cavities play an indispensable role in brain functioning. All these findings indicate an encouraging and promising direction in brain science research. However, it remains unclear today as how and in what pattern all such neuronal cliques and cavities are organized and mutually connected together. This calls for further endeavor into understanding the relationship between the complexity of higher-order topologies and the complexity of intrinsic neural functions of the brain. To do so, however, it needs to find most if not all cliques and especially cavities of different orders from the network.

Artificial intelligence, on the other hand, relies on artificial neural networks inspired by the brain neuronal network[10], including recurrent neural networks, convolutional neural networks, Hopfield neural network, etc. Now, given the recent

discover of higher-order cliques and cavities in the brain, the question is how to further develop artificial intelligence to an even higher level by utilizing the new knowledge about the brain topology. It is notable that a new neuronal network construction is recently proposed by an MIT research team inspired by the real structure of neuronal network of the C. elegans[11].

It is an important but challenging problem to understand how the brain store information, learn new knowledge and react to external stimuli, as well as its adaptively created topological connections and parallel computing patterns, which depend on in-depth studies of the brain neuronal network. Recently, the Brain Initiative project of USA[12], the Human Brain project of EU[13] and the China Brain project[14] are established to take such big challenges.

In retrospect, many innovative mathematicians contributed a lot of fundamental work to related subjects, such as Euler characteristic number, Betti number, the notions of groups introduced by Abel and Galois and higher-order Laplacian matrices as well as Euler-Poincaré formula and the homology group. This also demonstrates the importance of studying cliques and cavities for further development of network science. In addition, the advance from pairwise interactions to higher-order interactions in dynamics of complex systems requires the knowledge of higher-order cliques and cavities of networks[15]. The numbers of zero eigenvalues of higher-order Hodge-Laplacian matrices are equal to the corresponding Betti numbers, while their associate eigenvectors are closely related to higher-order cavities[16].

Motivated by all the above observations, this paper investigates the important issue of the computability of a complex network, based on which the study continues to find higher-order cliques and their Euler characteristic number, as well as higher-order Betti numbers and higher-order cavities. The approach starts from k-core decomposition[17], through finding cliques of different orders, and then performs a sequence of computations on the ranks of the corresponding boundary matrices to obtain the Betti numbers. To that end, an optimized algorithm is developed for finding higher-order cavities. Finally, the paper shows how to apply the optimized algorithm to the neuronal network of C. elegans from a dataset, and find its all cliques and some cavities of different orders.

Results

For computable undirected networks, the proposed approach is able to find all higher-order cliques, thereby obtaining the Euler characteristic number and all Betti numbers, as well as some cavities of different orders. These can provide global information for understanding and analyzing the relationships between topologies and functions of various complex networks such as brain neural networks.

1. Computable Networks

For undirected networks, the concept of clique in graph theory refers to a complete subgraph, e.g., a node is a 0-clique, an edge is a 1-clique, a triangle is a 2-cliques, etc. For example, it is easy to find all such cliques from the sample network shown in Figure 1.
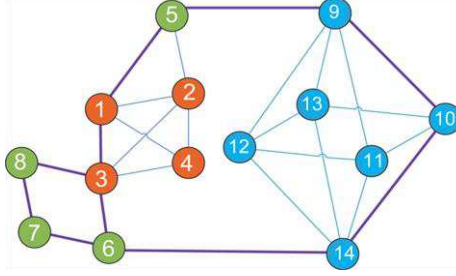

Figure 1. A sample network, with 14 nodes, 26 edges, 13 triangles, and 1 tetrahedron

For a given general large-scale complex network, however, finding all cliques of different orders is never an easy task. In fact, even just searching for a maximum clique (namely, a clique with the largest possible number of nodes) from a large network is a computationally NP-complete problem[18]. It is noticed that, to find all cliques of a large-scale undirected network, especially when the network is dense, the number of cliques are huge and will increase rapidly as the network size becomes larger. For example, in the real USair, Jazz and Yeast networks[19], if the number of cliques is limited to not more than 10^7 to be computable, the order of the cliques can go up only to 9, 6 and 4, respectively, as summarized in Table 1, where $|N|$ ($|E|$) is the number of nodes (edges). If the number of cliques does not decrease with the increase of the order, it will become impossible to compute them by using personal computers.

Table 1. Three real networks: their sizes and maximum cores k_{max},

maximum cliques c_{max} and the maximum order of the cliques when their numbers $< 10^7$

Network	$|N|$	$|E|$	k_{max}	c_{max}	$\max\{k|m_k <10^7\}$
USAir	332	2126	26	> 21	9 (m_9= 9121594)
Jazz	198	2742	29	29	6 (m_6= 2416059)
Yeast	2375	11693	40	> 30 ?	4 (m_4= 2454474)

For large and dense networks, k-core decomposition can determine the cells (layers), where the kth cell has all nodes with degrees at least k, and the kernel of the network has the largest core value, where nodes are very dense. Therefore, the largest core value k_{max} can be used to estimate the order of a maximum clique. For this reason, k-core decomposition is used to determine whether a given network is computable subject to the available limited computing resources. If the computing resources allow the number of cliques, with the first several lowest orders, be no more than 10^7, which

commercial laptops and PCs can handle, then the maximum core value should not be bigger than 30, say limited to $k_{max} = 25$, as detailed in Supplementary Information 1.

2. Clique-Searching Algorithm

The Bron–Kerbosch algorithm[20] is a popular scheme for finding all cliques of an undirected graph, while the Hasse-diagram algorithm[9] is useful for finding all cliques of a directed network. For computable networks, this paper proposes an algorithm for searching cliques, namely a common-neighbors scheme, which can quickly find all cliques of different orders and the associate Euler characteristic number.

For illustration, consider the sample network shown in Figure 1.

(1) Find all neighbors of each node, for which the index-number should be bigger than the index-numbers of the node (the shaded numbers are not so), as follows:

Note 1 {2,3,4,5}, Node 2 {1,3,4,5}, Node 3 {1,2,4,6,8}, Node 4 {1,2,3}, Node 5 {1,2}, Node 6 {3,7}, Node 7 {6,8}, Node 8 {3,7}, Node 9 {5,10,11,12,13}, Node 10 {9,11,13,14}, Node 11 {9,10,12,14}, Node 12 {9,11,13,14}, Node 13 {9,10,12,14}, Node 14 {10,11,12,13}.

Compute the number of nodes in 0-clique: $m_0 = 14$.

(2) Then, from the above list, generate edges in increasing order of node numbers:

(1,2)，(1,3)，(1,4)，(1,5)，(2,3)，(2,4)，(2,5)，(3,4)，(3,6)，(3,8)，(5,9)，(6,7)，(6,14)，(7,8)，(9,10)，(9,11)，(9,12)，(9,13)，(10,11)，(10,13)，(10,14)，(11,12)，(11,14)，(12,13)，(12,14)，(13,14).

Compute the number of edges in 1-clique: $m_1 = 26$.

(3) For every edge, check if its two nodes have common neighbors, for which the index-number should be bigger than the index-numbers of both nodes, and then record all such neighbors.

For example, edge (1,2) has common neighbors {3,4,5}, edge (1,3) has {4}, edge (2,3) has {4}, edge (9,10) has {11,13}, edge (9,11) has {12}, edge (9,12) has {13}, edge (10,11) has {14}, edge (10,13) has {14}, edge (11,12) has {14}, edge (12,13) has {14}.

However, edge (1,4) and edges (1,5), (3,4), (3,6), (3,8), (5,9), (6,7), (6,14), (7,8), (9,13), (10,14), (11,14), (12,14), (13,14) do not have any common neighbor.

Thus, the following triangles are obtained: (1,2,3), (1,2,4), (1,2,5), (1,3,4), (2,3,4), (9,10,11), (9,10,13), (9,11,12), (9,12,13), (10,11,14), (10,13,14), (11,12,14), (12,13,14).

Compute the number of triangles in 2-cliques: $m_2 = 13$.

(4) For each triangle, check if its three nodes have common neighbors (the index-number should be bigger than the index-numbers of three nodes), and record all

such neighbors.

Here, only triangle (1,2,3) has a common neighbor {4}, yielding 1 tetrahedron (1,2,3,4).

Compute the number of tetrahedrons in 3-cliques: $m_3 = 1$.

(5) This does not yield any more higher-order clique.

(6) Compute the Euler characteristic number[5]:

$$\chi = m_0 - m_1 + m_2 - m_3 = 14 - 26 + 13 - 1 = 0.$$

3. Computing Betti Numbers

Based on the above-obtained cliques of all orders, which can be used to generate boundary matrices B_k, $k = 1,2, ...$, where B_1 is the node-edge matrix, in which an element is 1 if the node is on the corresponding edge; otherwise, it is 0. Similarly, B_2 is the edge-face matrix, in which an element is 1 if the edge is on the corresponding face; otherwise, it is 0, etc. It is straightforward to compute the rank r_k matrices B_k for every $k = 1,2, ...$, using row-column operations in the binary field F_2, following the binary operation rules, namely $1 + 1 = 0$, $1 + 0 = 1$, $0 + 1 = 1$, $0 + 0 = 0$. Then, the Betti number[5] can be obtained as $\beta_k = m_k - r_k - r_{k+1}$.

One can also calculate the numbers of zero eigenvalues of higher-order Hodge-Laplacian matrices, so as to find the Betti numbers. To do so, it needs to follow some algebraic topology rules to form oriented cliques[16].
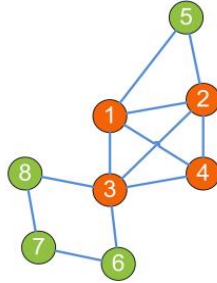

Figure 2. A subnetwork of the network shown in Figure 1.

As an example, consider the network shown in Figure 2, which is a subnetwork of the one shown in Figure 1, with the node-edge boundary matrix B_1 of rank $r_1 = 7$ as follows, where the shaded row is linearly dependent on the others:

B_1	(1,2)	(1,3)	(1,4)	(1,5)	(2,3)	(2,4)	(2,5)	(3,4)	(3,6)	(3,8)	(6,7)	(7,8)
1	1	1	1	1	0	0	0	0	0	0	0	0
2	1	0	0	0	1	1	1	0	0	0	0	0
3	0	1	0	0	1	0	0	1	1	1	0	0
4	0	0	1	0	0	1	0	1	0	0	0	0
5	0	0	0	1	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	1	0	1

Moreover, its edge-face boundary matrix of rank $r_2 = 4$ is obtained, as follows, where the shaded column is linearly dependent on the others:

$$
\begin{array}{c|ccccc}
B_2 & (1,2,3) & (1,2,4) & (1,2,5) & (1,3,4) & (2,3,4) \\
(1,2) & 1 & 1 & 1 & 0 & 0 \\
(1,3) & 1 & 0 & 0 & 1 & 0 \\
(1,4) & 0 & 1 & 0 & 1 & 0 \\
(1,5) & 0 & 0 & 1 & 0 & 0 \\
(2,3) & \mathbf{1} & 0 & 0 & 0 & 1 \\
(2,4) & 0 & \mathbf{1} & 0 & 0 & 1 \\
(2,5) & 0 & 0 & \mathbf{1} & 0 & 0 \\
(3,4) & 0 & 0 & 0 & \mathbf{1} & 1
\end{array}
$$

Table 2 summarizes all data for the network shown in Figure 1, in which the Euler characteristic number and Betti numbers satisfy the Euler-Poincaré formula[5]

$$\chi = \beta_0 - \beta_1 + \beta_2 - \beta_3 = 1 + 2 - 1 - 0 = 0$$

Table 2. Data for the network shown in Figure 1

Order k	0	1	2	3
m_k	14	26	13	1
r_k	0	13	11	1
$\beta_k = m_k - r_k - r_{k+1}$	1	2	1	0

4. Cavity-Searching Algorithm

The concept of cavity comes from the homology group in algebraic topology. Since a network usually has many 1-cycles, for instance the network shown in Figure 1 has nearly one hundred, to facilitate investigation they are classified into equivalent classes. In a network, each 1-cavity belongs to a linearly independent cycle-equivalent class[5] with the total number equal to the Betti number β_1. It is relatively easy to understand 1-cavity, which has boundary edges consisting of 1-cliques. It needs some imagination to understand higher-order cavities, which have boundary consisting of some cliques of the same order. So far, in the literature, only one 2-cavity consisting of 8 triangles is found and reported[8]. In the present paper, we found all possible smallest cavities and list them up to order 11 in Supplementary Information 2.

Since a cavity belongs to cycle-equivalent class, only one representative from the class with shortest length (namely, smallest number of cliques) is chosen for further discussion. To find the smallest one, however, optimization is needed.

4.1 Finding Cavity-Generating Cliques

Select a maximum linearly independent group of column vectors from the boundary matrix B_k as the minimum kth-order spanning tree, which consists of r_k

k-cliques, where r_k is the rank value of matrix B_k. Then, perform row-column binary operations to reduce it to a simplest form. In very row of the resultant matrix, the column index of the first nonzero element is used as the index of the k-clique in the spanning tree. For the example shown in Figure 2, those bold-faced 1's in the matrix B_1 correspond to columns (1, 2), (1, 3), (1, 4), (1, 5), (3, 6), (3, 8), (6, 7), which constitute a spanning tree. Note that the minimum kth-order spanning trees are not unique in general.

Then, find the maximum group of linearly independent column vectors from boundary matrix B_{k+1}, and obtain r_{k+1} ($k+1$)-cliques as a group of linearly independent cliques. From this group, search for a k-clique (the row index of the first nonzero element) that belongs to the boundary of the ($k+1$)-clique but does not belong to the kth-order spanning tree. In other words, the r_{k+1} k-cliques should not be a k-clique in the minimum spanning tree. If this cannot be found, then choose another maximum group of linearly independent column vectors from boundary matrix B_{k+1}, and try again. In this way, r_{k+1} ($k+1$)-cliques are found. As an example, see the example shown in Figure 2, where the bold-faced 1's in the boundary matrix B_2 correspond to the rows (2, 3), (2, 4), (2, 5), (3, 4), which are different from the cliques in the spanning tree.

Recall the formula of Betti numbers, $\beta_k = m_k - r_k - r_{k+1}$, which is the number of linearly independent k-cliques. Now, the task is to find the rest k-cliques that are not in the kth-order minimum spanning tree and also not on the boundaries of linearly independent $(k+1)$-cliques. These are called cavity-generating cliques. In the example shown in Figure 2, there is only one: (7, 8). On the minimum spanning tree, after including all linearly independent boundaries, adding every cavity-generating k-clique will create a linearly independent k-cavity, here the 1-cavity (3, 6, 7, 8).

4.2 Searching Cavities by 0-1 Programming

Every cavity-generating k-clique corresponds to at least one k-cavity. But, a cavity-generating k-clique may correspond to several different-length cavities, where the length is the number of cliques. Since a cavity is a linearly independent cycle with the smallest number of cliques, the task of searching for a cavity can be reformulated as a 0-1 programming problem.

Recall that there are m_k k-cliques, B_k is the boundary matrix between a $(k-1)$-clique and a k-clique, B_{k+1} is the boundary matrix between a k-clique and a $(k+1)$-clique, and a k-cavity consists of some k-cliques. Let C_k be the vector space based on k-cliques. A k-cavity can be expressed as $x = (x_1, x_2, \ldots, x_{m_k}) \in C_k$, in which each component x_i takes value 1 or 0, where 1 represents a k-clique with index i in the cavity, while 0 means no such cliques. Now, suppose that a

cavity-generating k-clique has index v among all k-cliques and let $e = (1, 1, \ldots, 1)^T$. Then, the problem of searching for a k-cavity becomes the following optimization problem to solve for a nonzero solution:

$$\min_{x \in C_k} f(x) = xe$$

s.t. (1) $x_v = 1$, (2) $B_k x^T = 0 \pmod 2$, (3) $\mathrm{rank}(x^T, B_{k+1})_{F_2} = 1 + r_{k+1}$.

Here, the first constraint means that the cavity comes from the cavity-generating k-clique with index v. The second constraint implies that the cavity is a k-cycle, namely the boundaries of k-cliques that form the cavity should appear in pairs. The third constraint shows that the k-cavity to be found is not a linear representation of the $(k + 1)$-cliques. This can avoid generating false cavities.

To ensure that the β_k cavities found are linearly independent, let the l-th k –cavity be $x^{(l)}$, and then complete the 0-1 programming: (1) $x_{v1}^{(l)} = 1$, $\cdots$, $x_{vl}^{(l)} = 1$; (2) $B_k x^T = 0 \pmod 2$; (3) $\mathrm{rank}(x^{(1)}, \cdots, x^{(l)}, B_{k+1})_{F_2} = l + r_{k+1}$ for $l = 1, 2, \ldots, \beta_k$.

It has been found that the sample network shown in Figure 1 has 2 1-cavities, where two cavity-generating 1-cliques are $x_{14} = 1$ corresponding to edge (7, 8) and $x_{11} = 1$ corresponding to edge (5, 9). Its optimization problem is as follows:

$$\min_{x \in C_1} f(x) = xe$$

s.t. (1) $x_{14} = 1$, (2) $B_1 x^T = 0 \pmod 2$, namely

$x_1 + x_2 + x_3 + x_4 = 0$, $x_1 + x_5 + x_6 + x_7 = 0$, $x_2 + x_5 + x_8 + x_9 + x_{10} = 0$, $x_3 + x_6 + x_8 = 0$, $x_4 + x_7 + x_{11} = 0$,

$x_9 + x_{12} + x_{13} = 0$, $x_{12} + x_{14} = 0$, $x_{10} + x_{14} = 0$, $x_{11} + x_{15} + x_{16} + x_{17} + x_{18} = 0$, $x_{15} + x_{19} + x_{20} + x_{21} = 0$,

$x_{16} + x_{19} + x_{22} + x_{23} = 0$, $x_{17} + x_{22} + x_{24} + x_{25} = 0$, $x_{18} + x_{20} + x_{24} + x_{26} = 0$, $x_{13} + x_{21} + x_{23} + x_{25} + x_{26} = 0$,

(3) $\mathrm{rank}(x^T, B_2)_{F_2} = 1 + r_2$.

Solving the above 0-1 programming problem, from $x_{14} = 1$ corresponding to (7, 8), it yields $x_{10} = 1$ corresponding to (3, 8), and from $x_{12} = 1$ corresponding to (6, 7), it yields $x_9 = 1$ corresponding to (3, 6), leading to the first cavity (3, 6, 7, 8). Then, replacing $x_{14} = 1$ by $x_{11} = 1$ yields the second cavity (1, 5, 9, 10, 14, 6, 3), which has 8 equal-length cavities, including 1-cavity (**2**, 5, 9, 10, 14, 6, 3) and 1-cavity (1, 5, 9, **11**, 14, 6, 3), etc. Finally, check the $\mathrm{rank}(x^{(1)}, \cdots, x^{(l)}, B_2)_{F_2} = l + r_2$ for $l = 1, 2$, which meet the requirements.

5. Cliques and Cavities of C. elegans

For a dataset of C. elegans with 297 neurons and 2148 synapses[21], all cliques and some cavities are obtained here by using the above-described techniques, which is compared to the typical random network (ER) model with the same numbers of nodes and edges. The results are shown in Figure 3 and Table 3.
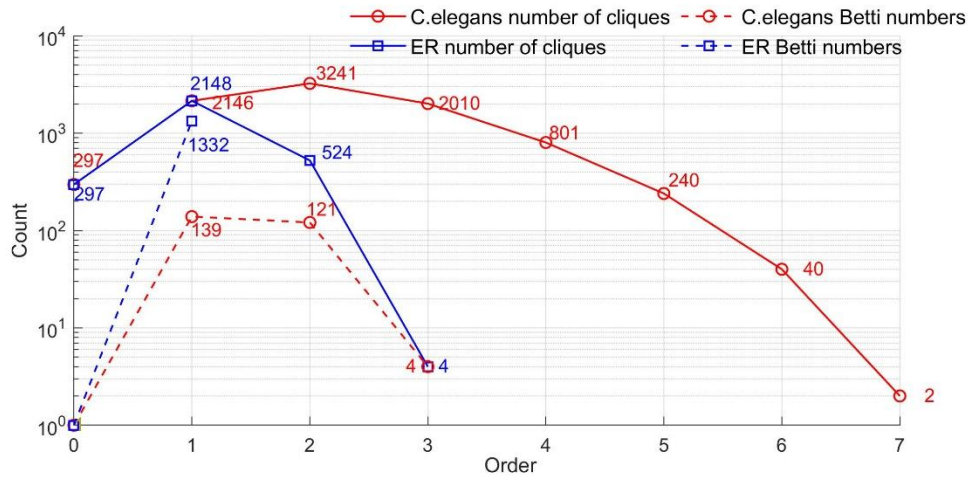
Figure 3. The number of cliques and the Betti numbers
for the C. elegans versus ER networks

Table 3. The Euler characteristic number, Betti numbers and the Euler-Poincaré formula

Network	The Euler characteristic number, Betti numbers and the Euler-Poincaré formula
C. elegans	$\chi = 297-2148+3241-2020+801-240+40-2 = 1-139+121-4 = -13$
ER	$\chi = 297-2148+524-4 = 1-1332 = -1331$

Since the highest-order nonzero Betti number is $\beta_3 = 4$, the C. elegans has 4 linearly independent 3-cavities, and these 4 cavities have cavity-generating 3-cliques (164, 163, 119, 118), (119, 167, 118, 227), (195, 185, 119, 118) and (227, 195, 119, 118), respectively. The cavity-generating 3-clique (164, 163, 119, 118) forms a 3-cavity with 8 nodes: (85, 13, 3, 164, 163, 119, 118, 158), it is the smallest 3-cavities, with structures[5] as shown in Figure 4 (a). The cavity-generating 3-clique (119, 167, 118, 227) forms a 3-cavity with 11 nodes: (163, 3, 162, 119, 154, 167, 118, 227, 85, 13, 164) as shown in Figure 4 (b). The cavity-generating 3-clique (195, 185, 119, 118) forms a 3-cavity with 8 nodes: (171, 13, 3, 195, 185, 119, 118, 173), as shown in Figure 4 (c). The cavity-generating 3-clique (227, 195, 119, 118) forms a 3-cavity with 8 nodes: (173, 13, 3, 227, 195, 119, 118, 185), as shown in Figure 4 (d), with details included in Supplementary Information, part 3.
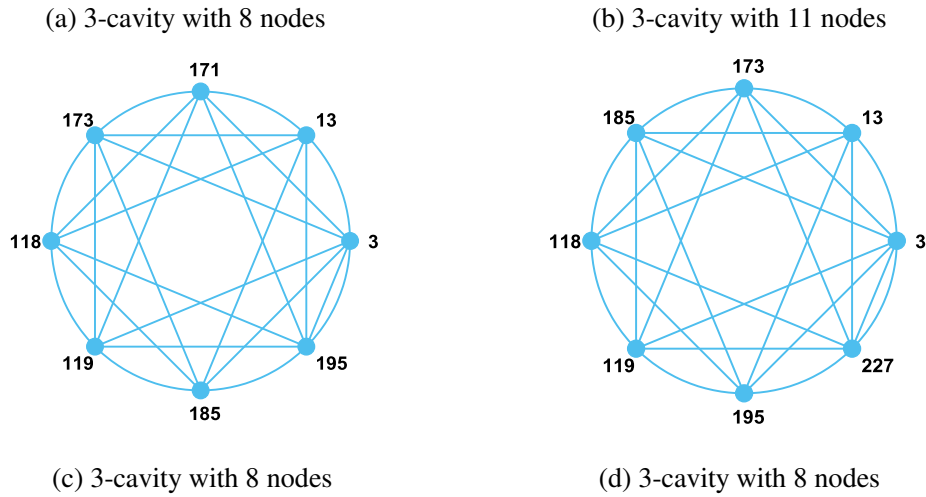
(a) 3-cavity with 8 nodes (b) 3-cavity with 11 nodes

(c) 3-cavity with 8 nodes (d) 3-cavity with 8 nodes
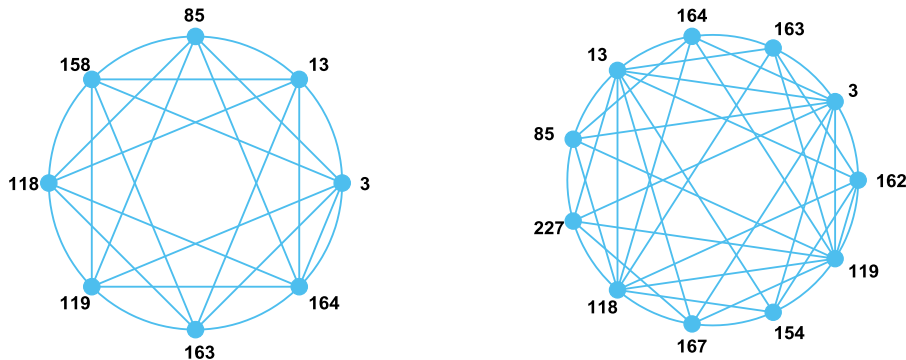
Figure 4. Four 3-cavities in the C. elegans neuronal network

Discussions

For a directed network, how to analyze higher-order cliques and cavities? In [9], by introducing directed cliques it develops a Hasse algorithm to find directed cliques. However, both concepts of cycle and especially cavity were not precisely defined therein. For an undirected network, the length of a cavity, namely the number of cliques that compose it, is longer than the length of the clique as a cycle having the same order with the cavity. For example, an undirected triangle of length 3 not only is a 2-clique but also is a 1-cycle, while 1-cavity at least is a quadrangle with length 4. For a directed network, however, this may not be true. For example, the smallest 1-cavity could be composed by two oppositely directed edges, with length 2, but a directed 2-clique could be a directed triangle of length 3. This implies the extreme complexity of directed cavities, which will be a topic for future investigation.

It should be noted that the key technique in the present approach is to examine various combinations of cliques and cavities, which differs from the focus on node degrees in the current investigation of complex networks, where the focus is on the statistical rather than topological properties. After comparing the neuronal network of the C. elegans to random network model, we found that they are very different regarding the numbers of cliques and cavities. From the perspective of brain science, various combinations of higher-order topological components such as cliques and cavities are of extreme importance, without which it is very difficult or even impossible to understand and explain the functional complexity of the brain. In fact, this seems provide reasonable supports to the recent works of many brain scientists.

The intrinsic combination of cliques and cavities also brings some unexpected problems in programming the algorithm. For example, because network spanning trees are not unique, the algorithm may not produce the required results when

searching for cavities. Efforts have been made to determine the information of cavities by eigenvectors corresponding to zero eigenvalues of higher-order Hodge-Laplacian matrices. However, similar non-uniqueness problem occurred in finding eigenvectors.

Method

Obviously, to solve the optimization problem of a k-cavity is difficult due to the third constraint therein. As a remedy, the optimization problem is separated into two parts. The first part is to use the following 0-1 programming problem to find a cycle:

$$\min_{x \in C_k} f(x) = xe \quad \text{s.t. (1)} \ x_v = 1, \ \text{(2)} \ B_k x^T = 0 \ (\text{mod } 2) \tag{1}$$

and the second part is to use the third constraint to check if the cycle is a cavity, i.e. to check $\text{rank}(x^{(1)}, \cdots, x^{(l)}, B_{k+1})_{F_2} = l + r_{k+1}$, for the l-th-cycle $x^{(l)}$, $l = 1, 2, \ldots, \beta_k$.

1. Searching a cycle

Because there is $B_k x^T = 0 \ (\text{mod } 2)$ in Eq. (1), it is not a traditional 0-1 linear programming problem. Let $\tilde{B}_k = [B_k, -2I]$ and $\tilde{x} = [x, y]$, where I is the identity matrix and $y = [y_1, \ldots, y_{m_k}]$. Then, $B_k x^T = 0 \ (\text{mod } 2)$ is equivalent to $\tilde{B}_k \tilde{x}^T = 0$. Since the minimum length of the k-cavity is $L_{\min} = 2^{k+1}$, Eq. (1) can be transformed to the following 0-1 linear programming problem:

$$\min xe \quad \text{s.t. (1)} \ xe \geq L_{\min}, \ \text{(2)} \ x_v = 1, \ \text{(2)} \ \tilde{B}_k \tilde{x}^T = 0, \ \text{(3)} \ x_i, y_i = 0 \ \text{or} \ 1 \tag{2}$$

Eq. (2) can be solved by using Matlab 0-1 linear programming toolbox, and the algorithm is described as follows.

Algorithm 1：Searching a cycle ($x^* = FindCycle(B_k, v, L_{\min})$)

Input：boundary matrix B_k

      index of cavity-generating clique v

      length of the smallest cycle $L_{\min}$

Output：cycle x^*

(1) $\tilde{B}_k \leftarrow [B_k, -2I]$

(2) Get x^* by using Matlab to solve the optimization Eq. (2)

2. Finding All Cavities

One needs to check the third constraint to see if the cycle found by Algorithm 1 is a cavity. If not, increase the length of the cycle 2^{k-1} cliques obtained by Algorithm 1 to search again for it. Thus, the algorithm for searching cavities is as follows:

Algorithm 2： Searching k-cavities $\left(\left\{\boldsymbol{x}_1^*,\cdots,\boldsymbol{x}_{\beta_k}^*\right\}=FindCavity\left(B_k,B_{k+1},\left\{v_1,\cdots,v_{\beta_k}\right\},k\right)\right)$

Input： boundary matrices B_k and B_{k+1}

indices of all cavity-generating cliques $\left\{v_1,\cdots,v_{\beta_k}\right\}$

Output： all k-cavities

(1) For $j \leftarrow 1\,to\,\beta_k$

(2) $L_{\min} \leftarrow 2^{k+1}$

(3) $\boldsymbol{x}^* \leftarrow FindCycle\left(B_k,v,L_{\min}\right)$

(4) While $rank\left(\left[\left(\boldsymbol{x}_1^*\right)^T,\cdots,\left(\boldsymbol{x}_{j-1}^*\right)^T,\left(\boldsymbol{x}^*\right)^T,B_{k+1}\right]\right) < rank\left(\left[\left(\boldsymbol{x}_1^*\right)^T,\cdots,\left(\boldsymbol{x}_{j-1}^*\right)^T,B_{k+1}\right]\right)+1$

(5) $L_{\min} \leftarrow L_{\min} + \left(\left\lfloor\dfrac{(\boldsymbol{x}^*\boldsymbol{e}-L_{\min})}{2^{k-1}}\right\rfloor+1\right)2^{k-1}$

(6) $\boldsymbol{x}^* \leftarrow FindCycle\left(B_k,v,L_{\min}\right)$

(7) If $\boldsymbol{x}^* == \varnothing$ （the optimization problem has no solution）

(8) Break

(9) End

(10) End

(11) $\boldsymbol{x}_j^* \leftarrow \boldsymbol{x}^*$

(12) End

Data availability

Data used in this work can be accessed at
http://linkprediction.org/index.php/link/resource/data/1

Code availability

The code for the numerical simulations presented in this article is available from the corresponding authors upon reasonable request.

References

1 Watts, DJ, Strogatz, SH. Collective dynamics of 'small-world' networks. 1998; *Nature* **393**: 440-442.

2 Erdös, P, Rényi, A. On random graphs. *Publicationes Mathematicae* 1959; **6**: 290-291.

3 Barabási, A-L, Albert, R. Emergence of scaling in random networks. 1999; *Science* **286**: 509-512.

4 Shi, DH, Chen, GR, Thong, WWK *et al*. Searching for optimal network topology with best possible synchronizability, *IEEE Circ. Syst. Magaz.* 2013; **13**: 66-75.

5 Shi, DH, Lü, LY, Chen, GR. Totally homogeneous networks. *Natl. Sci. Rev.* 2019; **6**: 962-969.

6 Zomorodian, A, Carlsson, G. Computing persistent homology. *Discrete Comput. Geom.* 2005; **33**: 249-274.

7 Gu, XF, Yau, ST. Computational Conformal Geometry -- Theory. *Int. Press of Boston,* 2008.

8   Sizemore, AE, Giusti, C, Kahn, A *et al*. Cliques and cavities in the human connectome. *J. Comput. Neurosci.* 2018; **44**: 115-145.

9   Reimann, MW, *et al*. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in Comput. Neurosci.* 2017; **11**: 00048.

10 Mohamad H. Hassoun, Fundamentals of Artificial Neural Networks. *MIT Press*, 1995.

11 Lechner, M, Hasani, R *et al*. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*. 2020; **2**: 542-652.

12 https://en.wikipedia.org/wiki/BRAIN_Initiative
    https://braininitiative.nih.gov/

13 https://en.wikipedia.org/wiki/Human_Brain_Project
    https://www.humanbrainproject.eu/en/

14 https://en.wikipedia.org/wiki/China_Brain_Project

15 Battiston, F, Latora, V, Petri, G *et al*. Networks beyond pairwise interactions: structure and dynamics, *Phys. Rep*. 2020; **05**: 004.

16 Millan, AP, Torres, JJ, Bianconi, G. Explosive higher-order dynamics on simplicial complexes. *Phys. Rev. Lett.* **124**: 218301. (2020)

17 Kitsak, M, Makse, HA, *et al*. Identification of influential spreaders in complex networks, *Nature Phys.*, 2010; **6**(11): 888-893.

18 Bomze, IM, Budinich, M, Pardalos, PM, Pelillo, M. The maximum clique problem. In Handbook of Combinatorial Optimization, pp. 1-74. *Springer*, Boston, MA, 1999.

19 Fan, TL, Lü, LY, Shi, DH, Zhou T. Characterizing cycle structure in complex networks. arXiv:2001.08541 [physics.soc-ph]

20 Bron, C, Kerbosch, J. Algorithm 457: Finding all cliques of an undirected graph, *Commun. ACM*, 1973; **16**(9): 575-577.

21 Rossi, RA, Ahned, NK. The network data repository with interactive graph analysis and visualization. In Twenty-Ninth AAAI Conference, *AAAI Press*, 2015; 4292-4293.

Acknowledgements

Author contributions

DS and GC developed the theory and wrote the text. ZC, XS, CM, YL and QC performed the simulations and computations for cross check. All authors checked and verified the entire manuscript.

Competing interests

Authors declare no competing interests.

Correspondence and requests for materials should be addressed to DS and GC.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SIComputingCliquesandCavitiesinNetworks.pdf