Scalable Brain Network Construction on White Matter Fibers

Moo K. Chung1,3,6*, Nagesh Adluru3, Kim M. Dalton3, Andrew L. Alexander2,3,5, Richard J. Davidson3,4,5

1Department of Biostatistics and Medical Informatics, 2Department of Medical Physics, 3Waisman Laboratory for Brain Imaging and Behavior 4Department of Psychology, 5Department of Psychiatry, University of Wisconsin, Madison 6Department of Brain and Cognitive Sciences, Seoul National University, Korea

ABSTRACT

DTI offers a unique opportunity to characterize the structural connectivity of the human brain non-invasively by tracing white matter fiber tracts. Whole brain tractography studies routinely generate up to half million tracts per brain, which serves as edges in an extremely large 3D graph with up to half million edges. Currently there is no agreed-upon method for constructing the brain structural network graphs out of large number of white matter tracts. In this paper, we present a scalable iterative framework called the $\epsilon$-neighbor method for building a network graph and apply it to testing abnormal connectivity in autism.

1. INTRODUCTION

Structural brain connectivity has been usually modeled as a network graph using white matter fiber tracts in DTI. The whole gray matter has been traditionally parcellated into $n$ disjoint regions. White matter fibers provide information of how one gray matter region is connected to another via a $n \times n$ connectivity matrix. The connectivity matrix is then thresholded to produce a binarized adjacency matrix, which is further used in constructing a graph with $n$ nodes. However, there is no gold standard for gray matter parcellation which makes the identification of node depend on the choice of parcellation. Depending on the scale of parcellation, the parameters of graph, which characterize graph topology, vary considerably up to 95%. Another problem of the parcellation is the arbitrariness of thresholding connectivity matrix. The topological parameters such as sparsity and clustering coefficients change substantially depending on the level of threshold.

The problems of parcellation and the subsequent arbitrary thresholding can be avoided if we do not use any parcellation in building the network. So the question is whether it is possible to construct a network graph without the usual parcellation scheme. This paper presents a novel network graph modeling technique called the $\epsilon$-neighbor construction that avoids parcellation and the subsequent thresholding of the connectivity matrix. Instead of using the pre-specified parcellation, we propose to use the two end points of fibers as network nodes while the fibers themselves serve as the edges connecting nodes.

The $\epsilon$-neighbor construction is motivated by the Rips complex of point cloud data, which has been used to characterize the topology of the point cloud data. The Rips complex is a graph constructed by connecting two data points if they are within specific distance $\epsilon$. The problem of the Rips complex is that given $n$ data points, it exactly produces a graph with $n$ nodes so the resulting graph becomes very complicated when $n$ becomes large. Unlike the Rips complex, the $\epsilon$-neighbor method does not use all data points in constructing a graph so it significantly reduces the complexity of data. Further, while the point cloud data does not have any hidden topological constraint, the two end points of white matter fibers are connected so we are actually dealing with paired point cloud data. So the $\epsilon$-neighbor construction is different from building the Rips complex and offers substantial computational advantage.

*Send correspondence to Moo K. Chung. 1500 Highland Ave. Madison, WI. 53705. USA. E-mail: mkchung@wisc.edu.
The DTI data from 31 subjects were used in this study: (i) 17 subjects with high functioning autism spectrum disorders (ii) 14 control subjects matched for age, handedness, IQ, and head size. The Autism Diagnostic Interview-Revised\(^{15}\) was used for diagnoses by trained researchers.\(^{3}\) Diffusion weighted images were acquired in 12 non-collinear diffusion encoding directions with diffusion weighting factor of $b=1000 \text{ s/mm}^2$ in addition to a single ($b=0$) reference image. Eddy current related distortion and head motion of each data set were corrected using AIR\(^{17}\) and distortions from field inhomogeneities were corrected using custom software algorithms based on.\(^{11}\) The six tensor elements were calculated using non-linear fitting methods\(^{1}\) We have used nonlinear tensor image registration algorithm given in\(^{19}\) for spatial normalization. This approach combines full tensor co-registration and high-dimensional diffeomorphic spatial normalization. The registration is based on an iterative strategy,\(^{12,19}\) where the initial template was computed as the average of original DTI. After DTI were aligned to a template space, we perform streamline based tractography using the TENsor Deflection (TEND) algorithm,\(^{2,13}\) Figure 1 shows the subsampled tractography result for a single subject.

### 3. $\epsilon$-NEIGHBOR CONSTRUCTION

Suppose a whole brain tractography result yields $k$ number of tracts. The $i$-th tract consists of two end points $e_{i1}$ and $e_{i2}$. In constructing the network graph, we only need to worry about the two end points of of a tract since all other points along the tract are connected to these two end points. The end points of tracts are considered as nodes of a graph while the tracts are considered as edges of the graph. The TEND algorithm can generate upward of half million tracts but it is not practical to construct a massive graph with half million edges. So we have developed a scalable iterative network graph construction technique called the $\epsilon$-neighbor construction.

Let $G_k = \{V_k, E_k\}$ be a 3D graph with vertex set $V_k$ and edge set $E_k$ at the $k$-th iteration. We define the distance $d(p, G_k)$ of a point $p$ to the graph $G_k$ to be the shortest Euclidean distance between $p$ and all points in $V_k$

$$d(p, G_k) = \min_{q \in V_k} \|p - q\|.$$  
We say point $p$ is the $\epsilon$-neighbor of graph $G_k$ if $d(p, G_k) \leq \epsilon$. The threshold $\epsilon$ will be called the resolution of graph and it determines the scale at which we construct the graph. Larger the value of $\epsilon$, cruder the constructed graph becomes with less number of nodes. Consider a tract with two end points $e_{11}$ and $e_{12}$. The algorithm then starts with the graph $G_1 = \{V_1, E_1\}$ with $V_1 = \{e_{11}, e_{12}\}$, $E_1 = \{e_{11}e_{12}\}$. For the edge set $E_1$, we simply denote the edge connecting two nodes $e_{11}$ and $e_{12}$ as $e_{11}e_{12}$. In the next iteration, we add the second tract with two end points $e_{21}, e_{22}$ to the existing graph $G_1$. There are four possibilities:
Figure 2. Example of constructing 3D network graph by adding one tract at a time to the existing graph. There are four possible cases of connecting the two end points (indexed 5 and 6) of the tract to the existing graph with nodes \( \{1, 2, 3, 4\} \). Depending on the proximity of \( \{5, 6\} \) to nodes \( \{1,2,3,4\} \), we either do nothing, or add connect one node (either 5 or 6) or add two nodes (both 5 and 6) to the graph.

1. \( e_{21} \) and \( e_{22} \) are all \( \epsilon \)-neighbors of \( G_1 \). Since the end points \( e_{21} \) and \( e_{22} \) are close to the already existing graph \( G_1 \), we do not change the vertex set, i.e. \( V_2 = V_1 \). Now check if the edge \( e_{21}e_{22} \) is in the edge set \( E_1 \) and add them if it is not found in the edge set. In this case we have \( E_2 = E_1 \cup \{e_{21}e_{22}\} \).

2. Only \( e_{21} \) is an \( \epsilon \)-neighbor. We only to add \( e_{22} \) to \( V_1 \) and let \( V_2 = V_1 \cup \{e_{21}\}, E_2 = E_1 \cup \{e_{21}e_{22}\} \).

3. Only \( e_{22} \) is an \( \epsilon \)-neighbor. We add \( e_{21} \) to \( V_1 \) and let \( V_2 = V_1 \cup \{e_{22}\}, E_2 = E_1 \cup \{e_{21}e_{22}\} \).

4. \( e_{21} \) and \( e_{22} \) are not \( \epsilon \)-neighbors. We add the end points to the vertex set and add the edge to the edge set. In this case, \( e_{21}e_{22} \) forms a disjoint edge and we have \( V_2 = V_1 \cup \{e_{21}, e_{22}\}, E_2 = E_1 \cup \{e_{21}e_{22}\} \).

The procedure is iteratively performed to every tracts until we exhaust all the tracts. The MATLAB code for performing \( \epsilon \)-neighbor construction is given in brainimaging.waisman.wisc.edu/~chung/graph.

4. CONNECTED COMPONENTS

The constructed 3D networks graph can be uniquely parameterized by transforming the graph into adjacent matrices. The adjacency matrix \( A = (a_{ij}) \) of a graph is constructed on the fly at each iteration by checking if we are adding a new edge to the existing edge set. If nodes \( i \) and \( j \) are connected, we let \( a_{ij} = 1 \) and \( a_{ij} = 0 \) otherwise. The adjacency matrix is symmetric. The adjacency matrix contains sufficient information to reconstruct the graph. For the subsequent analysis on connected components, 6mm-neighbor graph was used. This particular resolution is chosen since it is the largest integer resolution that produces the node numbers below 1000.

As an illustration of \( \epsilon \)-neighbor method, we characterize the abnormal brain network in autism using the connected components of a graph. The number of connected components of a graph is a topological invariant that measures the number of structurally independent or disjoint subnetworks. It can be interpreted as the zeroth Betti number \( \beta_0 \) in algebraic topology.\(^5\) The connected components can be identified using the Dulmage-Mendelsohn decomposition,\(^6\) which has been widely used for decomposing sparse matrices into block triangular forms in speeding up matrix operations. Figure 3 shows connected components in 4 brain networks. All nodes in the same connected component are colored identically. Most of nodes belong to the largest connected component indicating the brain network is highly connected. There are only 4% of nodes that are not connected to the largest connected component while the remaining 96% are all connected in average. We have plotted those 4% of nodes that are not part of the largest connected component for all subjects (Figure 4). Figure 4 shows the clustering pattern between the groups differs. Although these nodes are scattered in most parts of brain, high concentration of clustering occurs on the right parietal lobe for the control subjects. We tested if the size of the largest connected components differ between the groups. At 6mm resolution, control subjects have \( 642.86 \pm 68.60 \) nodes in the largest component while autistic subjects have \( 607.12 \pm 39.39 \) nodes. Note that we do not need to
account for brain size difference since networks constructed in the normalized space. The cluster size difference is significant ($p$-value = 0.079). We expect a larger sample size would increase the statistical significance.

5. $\epsilon$-FILTRATION

The analysis in the previous section characterizes networks at the last iteration of the $\epsilon$-neighbor construction. In this section, we present the idea of quantifying the network over all iterations via $\epsilon$-filtration, which is motivated by the Rips filtration in persistent homology.

For each given $\epsilon$, we have Rips complex $G_{\epsilon}$. By increasing the $\epsilon$ value, we have the Rips filtration, a sequence of larger Rips complexes:

$$G_{\epsilon_1} \subset G_{\epsilon_2} \subset G_{\epsilon_3} \subset \cdots$$

for $\epsilon_1 \leq \epsilon_2 \leq \epsilon_3 \leq \cdots$. During the Rips filtration, the topological features such as the Betti numbers change. The topological change over the filtration can be visualized by using the barcode. In the barcode, we plot the zeroth Betti number $\beta_0$ over the changing $\epsilon$ value. The resulting bar code is a decreasing function of $\epsilon$ and its decreasing pattern can be used to discriminate groups. Although the Rips filtration completely characterizes the topological change of a network, it is difficult to biologically interpret what it really means to have changing network over scale $\epsilon$. 
Similar to the Rips filtration, the $\epsilon$-filtration is a sequence of networks obtained from the $\epsilon$-neighbor method. At the $k$-th iteration, we have a network $G_k$. As the number of iterations increases, we are generating a sequence of larger networks

$$G_1 \subset G_2 \subset G_3 \subset \cdots,$$

which we will call the $\epsilon$-filtration. On the other hand, the $\epsilon$-filtration is much easier to interpret since it shows the actual process of network construction.

We computed the size of the largest component in each iteration in the $\epsilon$-filtration (Figure 5). It is always an increasing function and at about 6000 iterations, we begin to see the significant group difference in the increasing pattern. For instance, at 6000, 7000 and 8000 iterations, we have the $p$-values of 0.058, 0.038, 0.04 respectively. The control network (blue) is integrating into the largest component much faster than the autistic network (red). The growth of the size of the largest component is higher in controls. A schematic understanding of the underlying process is given in Figure 6. The control subjects are expected to have hub nodes that speed up the integration of disjoint components into the largest component.

6. CONCLUSION

We have presented a novel data-driven connectivity graph construction method for DTI. The proposed $\epsilon$-neighbor network construction can avoid the problem of parcellation and arbitrary connectivity matrix thresholding. The method is applied in showing abnormal connectivity pattern in autistic subjects in the $\epsilon$-filtration. The autistic network shows slower integration rate than the control network in the filtration demonstrating reduced network integration capability and efficiency.

REFERENCES


Figure 6. An example of how the size of connected component changes in the $\epsilon$-filtration. Originally a network consists of three disjoint components \{1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}. The size of the largest components is 3. In the next iteration, we connect the nodes 2 and 4 and obtain two components \{1, 2, 3, 4, 5\}, \{6, 7, 8\}. The size of the largest component is 5. In the 2nd iteration, we connect the nodes 2 and 7 and eventually obtain a giant component of size 8. The node 2 is more than a hub and connects all disjoint components together. By removing the node 2, the giant components disintegrate into three disjoint components. The sudden jumps in Figure 5 are showing the introduction of such nodes in the network.


