Approaches to Brain Parcellation using Energy Statistics and Graph Partitioning

Felix Xiao

Advisor: Professor Han Liu

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering
Department of Operations Research and Financial Engineering
Princeton University

June 2016

Abstract

We formulate the task of brain parcellation (dividing the brain into functionally homogenous regions) as a graph partitioning problem. We devise new model-free criteria for validating parcellations based on statistical dependency between parcels using distance correlation. Based off our criteria we pose a new graph cut-type problem called Max Average Within Edge (MAWE), wherein the objective is to maximize the sum for each component, of the average weight of all edges with both endpoints in the component.

In chapter 4, we propose a family of heuristic algorithms based on a new Contractible Graph data structure for attaining good solutions for MAWE and give the results of these algorithms on a resting state fMRI scan.

In chapter 5 we explore the well-established spectral ratio-cut minimization technique and measure its performance in MAWE on the same data set.

In chapter 6 we do the same for the more recently proposed nonnegative adjacency matrix factorization method, an alternative to spectral partitioning.

In chapter 7 we show that the MAWE problem can be reduced to a special instance of generalized 0-1 fractional programming which has an mixed integer programming equivalent.

Chapter 8 presents the results of these partitioning methods on 20 (?) autism and control fMRI scans.

Acknowledgements

Contents

List of Tables

List of Figures

Approaches to Brain Parcellation using Energy Statistics and Graph Partitioning

Felix Xiao

March 31, 2016

Contents

Chapter 1

Functional MRI Data and Brain Parcellation

Paragraph 1: Write here what would happen in the ideal world

Paragraph 2: Write here why we cannot reach this ideal easily

Paragraph 3: Write here what we do instead to overcome this problem

1.1 fMRI Background

Functional magnetic resonance imaging or functional MRI (fMRI) is a functional neuroimaging procedure using MRI technology that measures brain activity by detecting changes associated with blood flow. When an area of the brain is in use, blood flow to that region also increases. The primary form of fMRI uses the blood-oxygen-level dependent (BOLD) contrast. The fMRI concept builds on the earlier MRI scanning technology and the discovery of properties of oxygen-rich blood. The basis of MRI revolves around changing the magnetic moment of proton in hydrogen atoms abundantly found in the human body. MRI brain scans use a strong, permanent, static magnetic field to align nuclei in the brain region being studied. Another magnetic field, the gradient field, is then applied to spatially locate different nuclei. Finally, a radiofrequency (RF) pulse is played to kick the nuclei to higher magnetization levels, with the effect now depending on where they are located. When the RF field is removed, the nuclei go back to their original states, and the energy they emit is measured with a coil to recreate the positions of the nuclei. MRI thus provides a static structural view of brain matter. The central thrust behind fMRI was to extend MRI to capture functional changes in the brain caused by neuronal activity. Differences in magnetic properties between arterial (oxygen-rich) and venous (oxygen-poor) blood provided this link.

The fMRI machine records the relative change detected within a time frame (typically 2 seconds) in magnetization as a 3-dimensional image. The 3-dimensionsal image it provides is built up in units called voxels. Each one represents a tidy cube of brain tissue – a 3-D image building block analogous to the 2-D pixel of computers screens, televisions or digital cameras. When visualized on a monitor, typically each intensity level is encoded as a varying shade of gray. {Probably good to include an image of the "typical" fMRI scan} Each voxel can represent a million or so brain cells. As measurements are made over time (for example, over 10 minutes), the neuroscietist collects multiple 3-dimensional images in the form of a time series.

{This information was primarily taken from wiki, https://www.youtube.com/watch?v=djAxjtN_
http://science.howstuffworks.com/mri.htm and http://www.howequipmentworks.com/mri_basics/.}

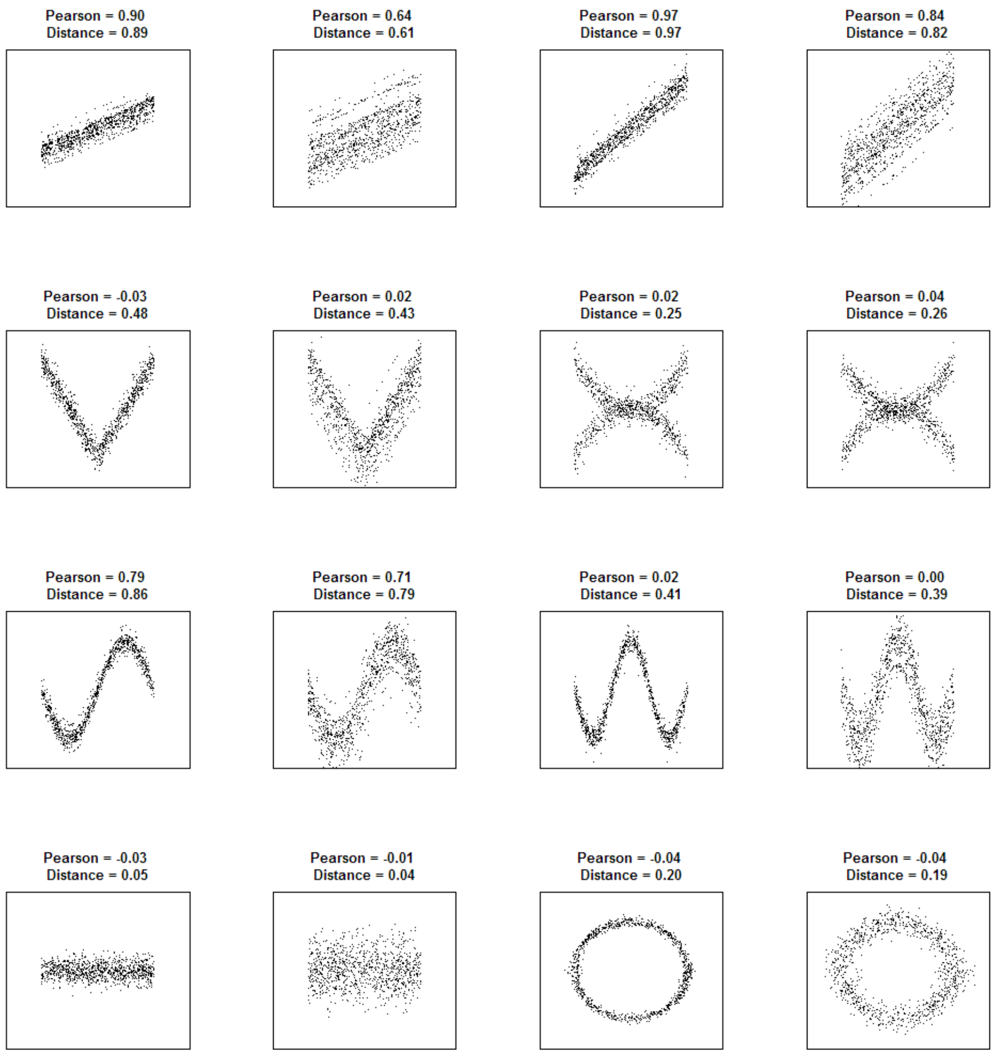
1.2 Parcellation Background

talk about AAL and do the literature review here.

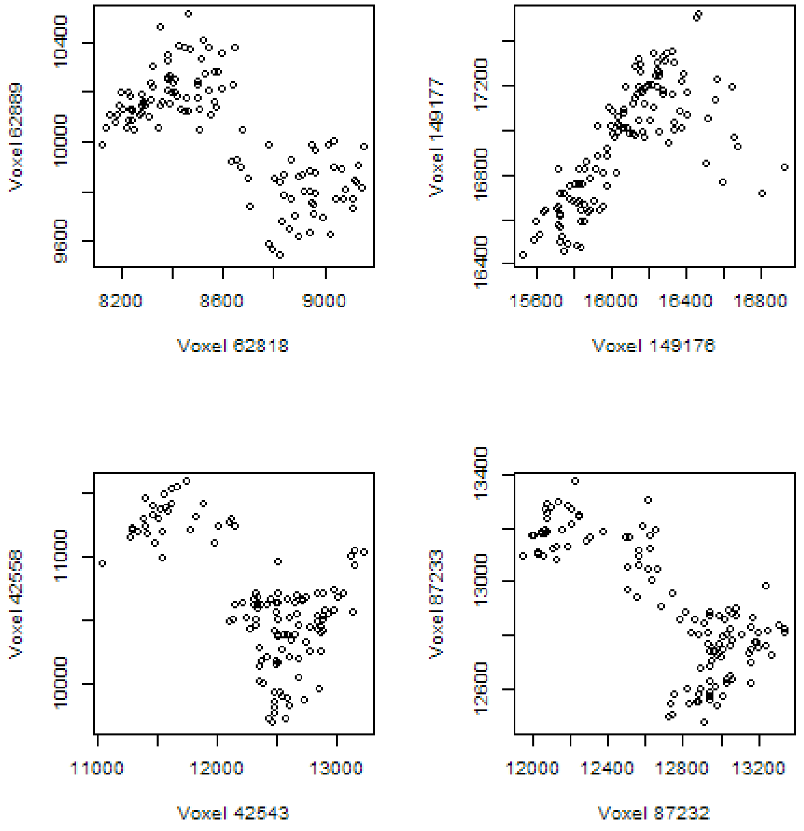
Functional parcellation of the human brain can be defined as the problem of partitioning the voxels into k disjoint connected components with the goal that the voxels within each component are, in a rough sense, s̈imilarẗo each other and voxels in different components are less ̈similar ̇. Such similarity has been defined in a multitude of ways in the literature [see lit review section ...]. For this project thus far I have taken similarity between voxels to mean statistical dependence.

1.3 Overall Strategy

Let's move this stuff about correlation to the beginning of the next chapter about energy statistics. For now, it's good enough to say in one paragraph that we'll use some nonlinear dependency in our procedure or something (high level summary) To measure dependence, statisticians have traditionally used the Pearson correlation coefficient, in addition to the rank-based Kendall tau and Spearman rho. These statistics work well when the underlying relationship between the two random variables is linear, in the case of Pearson, or can be linear after a monotonic transformation, in the case of Kendall and Spearman. Due to their restrictions, these correlation coefficients will fail to capture many kinds of dependency relationships. The figure below illustrates several instances of pairs of random variables whose depencency structure is not detected by the three correlation coefficients.



Non-linear dependency relationships also exist in the ABIDE 50002 fMRI data. The scatterplots below show time samples of spatially adjacent voxels. These instances were found by searching for the maximum difference in rank of energy distance correlation and the coefficient of determination, or Pearson squared.



Many studies on functional parcellation (Craddock 2012; Bellec 2006; Heller 2006) use Pearson's coefficient as the similarity measure between nearby voxels. Apart from underestimating the important of non-linear relationships, this method also distinguishes positive, upward-sloping correlation from negative. As a result in many of the edges between different parcels, the corresponding voxels would be strongly dependent with negative correlation.

1.4 About the Data

Autism spectrum disorders (ASD) represent a formidable challenge for psychiatry and neuroscience because of their high prevalence, lifelong nature, complexity and substantial heterogeneity. Roughly 1% of children worldwide are diagnosed with ASD centers2010autism. We approach the parcellation problem by using the Autism Brain Imaging Data Exchange (ABIDE) – a consortium aggregating and openly sharing 1112 existing resting-state functional magnetic resonance imaging (R-fMRI) data sets with corresponding structural MRI and phenotypic information from 539 individuals with ASDs and 573 age-matched typical controls (TCs; 764 years) .

For simplicity in this thesis, we focus {on NUMBER OF SUBJECTS YOU'RE ACTUALLY USING} scanned at the University of Pittsburgh School of Medicine. These autistic subjects included individuals from 7 to 35 years of age, with a well-characterized Autistic Disorder. The Autism Diagnostic Interview-Revised lord1994autism and the Autism Diagnostic Observation Schedule-General lord2000autism, as well as expert clinical opinion, were used to diagnose autism. Typical controls were healthy individuals, with no history of head trauma, birth complications, seizures, or psychiatric disorder. TC matched individually to the participants with autism on age (within 1.5 y in children, 3.5 y in adults), full-scale IQ (within 12 points) and gender. Individuals with Autistic Disorder were referred from the Center for Excellence in Autism Research (CEFAR) and the Autism Center of Excellence (ACE). TC were recruited from previous studies at the LNCD or by fliers and announcements. {If you want to know, I got this information from http://fcon$_1$000$.projects.nitrc.org/indi/abide/.Youproba$

In order to register each subject's brain to a common brain space, we use the Montreal Neurological Institute's standardized brain evans19933d,collins1994automatic. The MNI wanted to define a brain that is more representative of the population. They created a new template that was approximately matched to the Talairach brain in a two-stage procedure. First, they took 241 normal MRI scans, and manually defined various landmarks, in order to identify a line very similar to the AC-PC line, and the edges of the brain. Each brain was scaled to match the landmarks to equivalent positions on the Talairach atlas. They then took 305 normal MRI scans (all right handed, 239 M, 66 F, age 23.4 +/- 4.1), and used an automated 9 parameter linear algorithm to match the brains to the average of the 241 brains that had been matched to the Talairach atlas. From this they generated an average of 305 brain scans thus transformed - the MNI305. The current standard MNI template is the ICBM152, which is the average of 152 normal MRI scans that have been matched to the MNI305 using a 9 parameter affine transform. The International Consortium for Brain Mapping adopted this, the MNI152, as their standard template, and this is the template we will be using. We use the unsymmetrical MRI scan from MNI152 corresponding to voxels corresponding to cubes with an edge-length of 2 millimeters. This MNI152 template includes {WHAT ARE THE DIMENSIONS? I'll look this up lol.}.

We preprocessed the raw data from ABIDE using the Configurable Pipeline for the Analysis of Connectomes (C-PAC) alpha version 0.3.9. C-PAC is an open-source software pipeline for automated preprocessing and analysis of resting-state fMRI data. The image preprocessing steps included slice-timing and motion correction based on the Friston Model, nuisance signal regression (including 5 CompCorr signals, the cerebrospinal fluid (CSF), motion and the global, linear, and quadratic signals) and temporal filtering (0.001-0.08Hz). The derived R-fMRI measures were normalized to Montreal Neurological Institute (MNI152) stereostatic space (2mm^3 isotropic) with linear regressions and spatially smoothed (applied FWHM = 6mm).

We then converted the 4-dimensional fMRI data (i.e., a 3-dimensional image varying with time) in a 2-dimensional matrix whereby each column represents a different voxel and each row represents a different sample from a different time. Since C-PAC removed most autocorrelations in the data, we can reasonably treat each observation (i.e, each row) as drawn from the same unknown distribution.

In this investigation, all parcellation and validation procedures were conducted on the ABIDE 50002 fMRI data set. This data set contains 233305 voxels and 124 time samples. Spatial information is encoded as a graph; each voxel is represented by a vertex, and each vertex has up to 6 edges connecting the voxel to its cubically adjacent neighbors. The weights on the edges are sample energy distance correlations between the two connected voxels (Szekely 2013).

1.5 Notation

If you don't have any, you can omit this section :p

1.6 Chapter Summaries

Put the last part of what was currently in your abstract here.

Chapter 2

Criteria for Evaluating Parcellations

In chapter one we discussed our graphical approach to the brain parcellation problem. We construct a weighted undirected graph where each vertex is a voxel. The graph reflects the spatial position of the voxels; it connects each vertex to the vertices representing the voxel's six cubically adjacent neighbors. The weights on these edges are sample energy distance correlation statistics between the adjacent voxels in the time series and they measure statistical dependence between the voxels. Let $G(V, E)$ denote this graph, its vertices, and its edges.

In this context, a valid k-fold partition $\mathcal{P}_k$ of the graph G is a collection of vertex subsets $(V_1, ..., V_k)$ satisfying the following:

1. $V_i \neq \varnothing$ for all $V_i \in \mathcal{P}_k$
2. $\bigcup\limits_{i=1}^{k} V_i = V$
3. $V_i \cap V_j = \varnothing$ for all $V_i, V_j \in \mathcal{P}_k$
4. V_i is connected (i.e. for every two vertices in V_i, there is a path between them) for all $V_i \in \mathcal{P}_k$

In this chapter we will suggest various criteria for measuring the goodness-of-fit of partitions and discuss their statistical and computational advantages and drawbacks.

2.1 Within-Parcel Similarity

Voxels in the same parcel are ideally highly dependent on one another in the time series of fMRI data. As discussed in the previous chapter, distance correlation is a good measure of dependence. The distance correlation between two random vectors equals zero if and only if the two random vectors are independent, which is not true of correlation statistics such as Pearson's.

Let $\mathcal{R}(x, y)$ denote the distance correlation between two voxels x and y. Let V and W be any parcels. We will use E_V to denote the set of edges with one end in V and one end not in V, and $E_{V,W}$ the set of edges with one end in V and one in W.

Definition 2.1.1 (Within-Score)

$$\frac{1}{k} \sum_{V \in \mathcal{P}_k} \frac{1}{|V|^2} \sum_{x,y \in V} \mathcal{R}(x, y)$$

The Within-Score is non-spatial; it considers all pairs of voxels equally regardless of whether they are adjacent. As a result, it is a good measure of how much the voxels within each parcel are dependent on each other as a set. The downside of this criterion is that it is very expensive to compute. With over 300,000 voxels in an fMRI data set we would potentially have to compute tens of billions of distance correlation statistics, each of which takes time proportional to the number of samples squared.

An alternative and far less expensive criterion that measures within- parcel similarity works by counting distance correlations between adjacent pairs of voxels.

Definition 2.1.2 (Adjacent-Score)

$$\frac{1}{k} \sum_{V \in \mathcal{P}_k} \frac{1}{|E_{V,V}|} \sum_{(x,y) \in E_{V,V}} \mathcal{R}(x, y)$$

Rather than treat parcels as sets with no spatial information, the Adjacent-Score does the opposite by only considering the pairwise dependency of adjacent voxels. For sparse graphs such as ours, the number of distance correlation computations is proportional to the number of vertices.

An intermediate possibility we did not explore is to consider all pairs of voxels up to some maximum spatial distance from each other and perform a weighted averaging of sample pairwise distance correlations, with weights that depend on spatial distance.

2.2 Between-Parcel Dissimilarity

To evaluate parcellation quality, it is also useful to measure how dependent voxels belonging to different parcels are on each other. To this end we define two criterion similar to the Within-Parcel criterion; a non-spatial metric called the Between-Score and its spatial metric the Boundary-Score.

Definition 2.2.1 (Between-Score)

$$\frac{1}{\binom{k}{2}} \sum_{V,W \in \mathcal{P}_k, V \neq W} \frac{1}{|V||W|} \sum_{x \in V, y \in W} \mathcal{R}(x, y)$$

Definition 2.2.2 (Boundary-Score)

$$\frac{1}{\binom{k}{2}} \sum_{V,W \in \mathcal{P}_k, V \neq W} \frac{1}{|V||W|} \sum_{(x,y) \in E_{V,W}} \mathcal{R}(x, y)$$

Generally both of these quantities are more expensive to compute than their Within-Parcel counterparts. Boundary-Score is easy enough to compute for validation purposes, but does not convey much additional information beyond what the Adjacency-Score does, in the sense that the edges used in the computation of Adjacency-Score are the complement of the edges used in the Boundary-Score.

The ability of distance correlation to generalize to pairs of random vectors of arbitrary dimension gives us another way of computing the dependency between two parcels. The Multivariate Between-Score defined below treats parcels as random vectors and computes the distance correlation at the parcel level rather than voxel level. The result is a measure of non-spatial between-parcel similarity that is also computationally feasible. For this reason we will use Multivariate Between-Score as our primary measure of parcel dissimilarity.

Definition 2.2.3 (Multivariate Between-Score)

$$\frac{1}{\binom{k}{2}} \sum_{V,W \in \mathcal{P}_k, V \neq W} \mathcal{R}(V, W)$$

2.3 Graph Cuts

Closely related to the Boundary-Score is the notion of a graph cut from computer science. A *cut set* is the set of edges with endpoints in different parcels. The *cut weight* is the sum of weights of all edges in the cut set and can be expressed as

$$\frac{1}{2} \sum_{V \in \mathcal{P}_k} \sum_{x,y \in E_V} \mathcal{R}(x, y)$$

The *ratio cut* defined below is a weighted version of the cut weight:

$$\frac{1}{2} \sum_{V \in \mathcal{P}_k} \frac{1}{|V|} \sum_{x,y \in E_V} \mathcal{R}(x, y)$$

The subfield of graph partitioning is concerned with minimizing cut weight, ratio cut, and several other related quantities. Over the last several decades a number of highly effective approximation algorithms have been developed to find partitions of graphs that minimize these quantities. Later chapters will explore how these methods work for brain parcellation.

2.4 Balance and Jaggedness

In addition to the above distance correlation based criteria, there are two additional metrics concerned with parcel shape.

Definition 2.4.1 (Balance)

$$\frac{1}{k}\frac{1}{\max_{V\in\mathcal{P}_k}|V|}\sum_{V\in\mathcal{P}_k}|V|$$

The Balance-Score ranges from 1 (all equally sized parcels) to 0 (two parcels with one of size zero).

Definition 2.4.2 (Jaggedness)

$$\frac{1}{k}\sum_{V\in\mathcal{P}_k}\frac{|E_V|^{\frac{3}{2}}}{|V|}$$

The $\frac{3}{2}$ power makes the ratio invariant on parcel size. For instance, a $n \times n \times n$ cube of vertices would have a compactness of $6^{\frac{3}{2}}$.

2.5 Comparing Multiple Parcellations

Chapter 3

Local Search and Graph Growing Heuristics

We introduce several algorithms for generating brain parcellations. The algorithms in this chapter are all local search heuristics; they begin with n unconnected vertices and iteratively join adjacent ones into components until some stopping criterion is met.

For each algorithm, the resulting parcellation is presented, discussed, and evaluated according to the criteria introduced in the previous chapter.

3.1 The Add-Edge Algorithm

The first and simplest algorithm starts with an empty graph of n vertices and sequentially adds edges between adjacent voxels in order of highest sample distance correlation, until the graph has some prespecified number of connected components k. We will refer to this algorithm as "Unconstrained Add-Edge".

The Unconstrained Add-Edge algorithm produces severely imbalanced parcellations. In the 100-component graph, there was one component containing over 99.9% of all the vertices in the graph.

Our attempt to address this problem was to impose a filter on each edge considered, adding the edge only if at least one of the two following conditions are met:

1. At least one of the two components bridge by the edge is of size less than some prespecified parameter $s_{\min}$.
2. The union of the two components is of size $\leq s_{\max}$.

The restriction on adding new edges was not successful in creating balanced partitions.

3.1.1 Implementation of Unconstrained Version

A naive implementation of Unconstrained Add-Edge would re-compute the number of connected components in the graph (using linear-time bread-first or depth-first search) after each addition of an edge, resulting in a costly $O(EN)$ time complexity. A more efficient implementation takes advantage of the fact that each addition of an edge decreases the number of components in the graph by at most 1. Hence the algorithm needs only to compute the number of connected components after adding $c - k$ edges, where c is the current number of connected components of the graph, beginning at n.

Another implementation uses a binary search-type strategy and is $O((n+E)\log E)$. The idea is to "search" for the last edge to add to the graph by maintaining a range of possible last edges. In each iteration, the algorithm would add to the graph edges 1 to the midpoint of this range, compute the number of connected components, and adjust the range based on whether the number of components is higher or lower than the target K.

3.1.2 Implementation of Size-Constrained Version using Union-Find

The naive implementation must use BFS/DFS in each iteration to compute the sizes of the two components to be connected by an edge, and hence must have time complexity $O(EN)$. Fortunately, there is a way to sublinearly update information on the components of the graph, using the union-find data structure.

The core Union-Find data structure begins with an empty graph of N vertices and supports two operations. union(i, j) adds an edge between vertices i and j. root(i) returns an identifier for the component to which vertex i belongs. All vertices in the same component have the same root. We modified Union-Find to support an additional operation. component_size(i) returns the number of vertices belonging to the component containing i.

Union-Find represents each component as a rooted tree, with vertices in the graph mapping to nodes in the tree. Information about the tree is stored in two arrays of length N, parent and size, which are subject to the following invariants.

1. For each node i, parent[i] = node i's parent on the tree, unless i is a root node. If i is a root node, then parent[i] = i.

2. Nodes i and j are in the same component if and only if they are in the same tree, if and only if they share the same root node.

3. If i is a root node, then size[i] = the size of the component, or the number of nodes in the tree. If i is not a root node, then size[i] can be anything.

A baseline implementation of the three functions is

In addition to the baseline code above, there are two important optimizations:

1. Weighted union maintains information of the sizes of each component so that the root of the smaller component always becomes a child of the larger component's

Algorithm 1 Union-Find

```
function ROOT(i)
    while parent[i] $\neq$ i do
        i $\leftarrow$ parent[i]
    end while
    return i
end function

function UNION(i, j)
    parent[root(j)] $\leftarrow$ root(i)
end function

function COMPONENT_SIZE(i)
    return size[root(i)]
end function
```

root.

2. Path compression flattens the tree with each call to root. Specifically, when root is called on node i, each node traversed from i to the root has its parent set to be the root.

With these two optimizations, the time complexity of root, union, and component_size has been shown to be at least as good as $O(\log^* N)$ where $\log^*$ is the iterated logarithm, defined as the number of times the natural log must be applied to N so that it becomes less than or equal to 1.

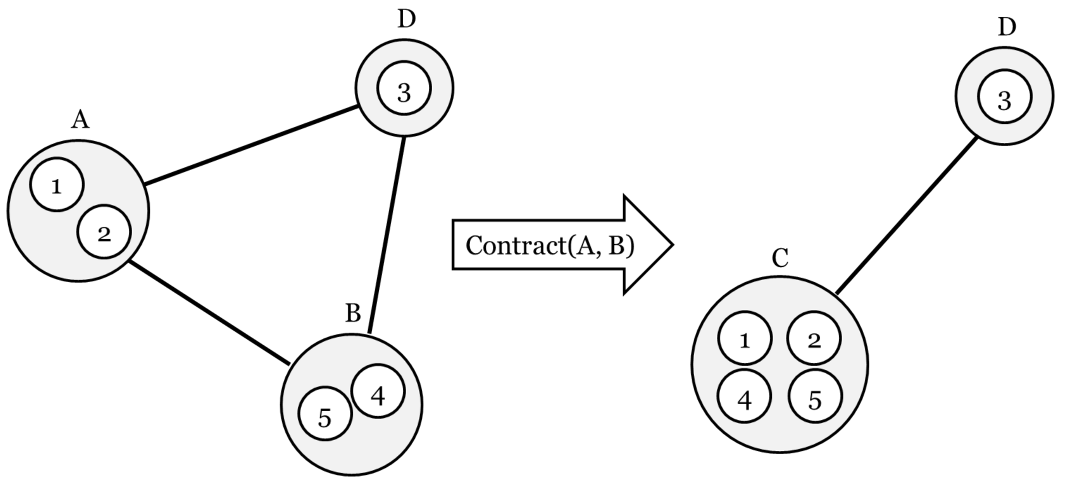
3.2 The Edge-Contraction Algorithm and Contractible Graphs

We propose a new data structure called the *Contractible Graph* (CG) for brain parcellation. The rationale behind the CG is a heuristic procedure for partitioning a graph into somewhat balanced components so as to maximize the Adjacent-Score (1.1.2).

The CG is a mapping of the vertices of the original graph to the vertices of a new graph. The vertices of the CG are called *components* and between any two components there exists exactly one weighted edge, henceforth called a *link*. The weight of a link $w_{A,B}$ between two components A and B in the CG equals the average weight of all edges in the original graph between vertices mapped to A and vertices mapped to B. If no such edges exist, the weight of the link is 0. Formally,

$$E_{A,B} = \{(i,j) \in E : i \in A, j \in B\}$$

$$w_{A,B} = \begin{cases} \frac{1}{|E_{A,B}|} \sum_{(i,j)\in E_{A,B}} w_{ij} & \text{if } |E_{A,B}| > 0 \\ 0 & \text{otherwise} \end{cases}$$



We say an edge (i, j) is *between* components A and B if i is in one of A or B and j is in the other. The *size* of a component is the number of vertices it contains. A *contraction* of a link (A, B) in a CG replaces components A and B with a new component (call it C) containing all vertices mapped to A or B, as illustrated in the figure above. Component C has one link to every other component in the CG, whose weights are the mean of the weights of the corresponding vertex edges, or 0 if no edge exists. Thus the contraction operation maintains the link-invariant property of CG. This leads to the Edge-Contraction algorithm, which begins with the original graph with all vertices as singleton components and contracts edges in a certain order until the graph has only k components in all.

Algorithm 2 Edge-Contraction

```
**Input:** Undirected positive-weighted graph $G$ and target component number $k$
Create a CG from $G$ so that every vertex maps to a unique component
**repeat**
    $\mathcal{S} \leftarrow$ smallest component(s) in the CG
    $(A, B) \leftarrow \underset{A \in \mathcal{S}}{\operatorname{argmax}}\, w(A, B)$
    Contract $(A, B)$
**until** CG has $k$ components
**Output:** Components of CG
```

Why does Edge-Contraction work better than the previous algorithms? The Edge-Contraction algorithm attempts to address two problems of the Size-Constrained Add-Edge algorithm: unbalanced parcels and poor Adjacent-Score relative to randomized graph. It solves the former problem by prioritizing edges that are connected to small components.

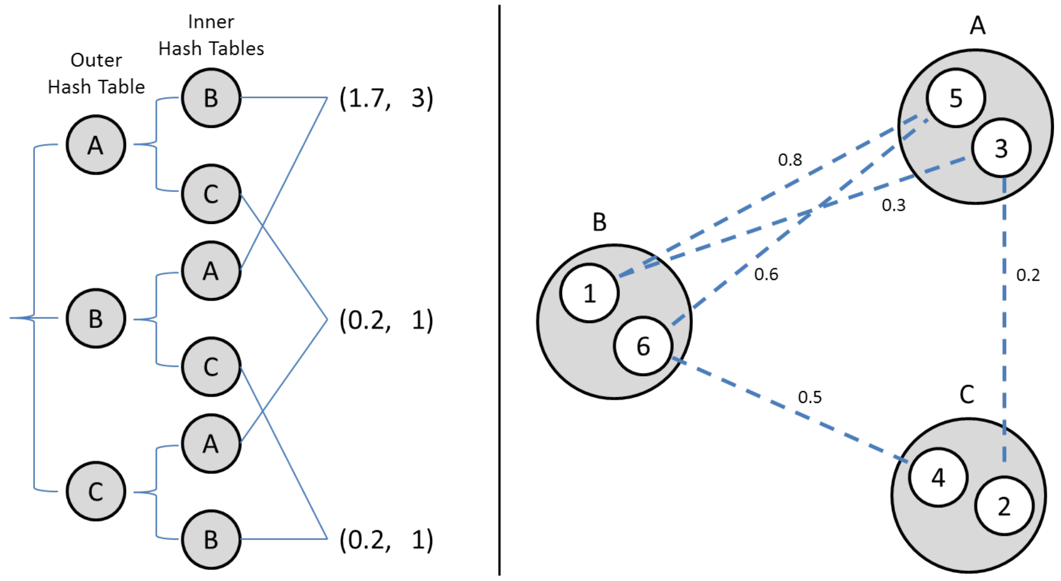
The low Adjacent-Score is likely due to the following scenario: when a vertex is added to a component, it might have multiple edges to that component. One edge might have a very high weight; this is the one that is officially “added”. However, the other edges with far lower weights are implicitly added as well, lowering the average edge weights within the component.

The Edge-Contraction algorithm handles this issue by maintaining that there can be at most one edge between any two components A and B, and further that the weight on such an edge is the mean of the weights on all edges that connect a vertex in A with a vertex in B.

3.2.1 Implementation using Nested Hash Tables and Priority Queue

In a Contractible Graph, the weight of the link between two components depends on the summed weight of all edges between them, and the number of such edges.

Our implementation of the CG uses *nested hash tables*, diagrammed below. The outer hash table maps each component A to an inner hash table, which maps all components B with a positive link to A to 1) the summed weights of the edges and 2) the number of edges between A and B.



Implementing the contraction of components B and C into a new component D on this nested hash table requires the following steps. The time complexity is stated assuming no hash collisions.

1. Compute $\mathcal{X}$, the set of all components that either B or C is linked to. $O(|E_B| + |E_C|)$

2. Create a new element in the outer hash table, D, and associate it with an empty inner hash table. $O(1)$

3. For each component $X \in \mathcal{X}$,

    - Retrieve $W(X, B) + W(X, C)$, the summed weights all edges between X and B and between X and C, and $|E_{X,B}| + |E_{X,C}|$, the number of such edges. These quantities are stored explicitly as a values in the inner hash table, so this operations is $O(1)$.

- Add a new component name D to the inner hash table of X and map it to $\big(W(X,B)+W(X,C), |E_{X,B}|+|E_{X,C}|\big)$. Delete elements B and C from the inner list of X. $O(1)$
- In the D inner hash table, add component name X and map it to to the same $\big(W(X,B)+W(X,C), |E_{X,B}|+|E_{X,C}|\big)$. $O(1)$

4. Delete B and C from the outer list.

Having described the contraction step, we will next discuss how to efficiently locate the link to be contracted. In computer science, a *Maximum Priority Queue* (MaxPQ) data type is a set of well-ordered objects that supports the following operations:

- *add(obj)*: Adds an object to the set.
- *remove_maximum()*: Removes and returns an object with the largest priority in the set.

Using the heap data structure, the above two operations both run in $O(\log n)$ time.

Each component on the CG will be associated with an element of the priority queue. The priority of component A is defined as

$$\max_X \; w_{A,X} - |A|$$

Since our graph link and edge weights are all between 0 and 1, the highest priority element in the queue always has the smallest size. Therefore, if the priority queue is up-to-date with the CG, the next link to be contracted according to Edge-Contraction has an endpoint component whose priority is the highest in the queue.

However, a complication arises from the fact that a contraction can change the priorities of components neighboring the contracting components, thereby making

Table 3.1: Results of Edge-Contract for Different Component Numbers

Number of Components	Adjacency	Multi-Boundary	Jaggedness	Balance
500	0.7991	0.7578	54.65	0.327
400	0.7990	0.7723	59.20	0.343
300	0.7974	0.7921	65.20	0.297
250	0.7955	0.7991	69.51	0.356
200	0.7941	0.8101	75.47	0.357
150	0.7931	0.8225	83.88	0.418
116	0.7913	0.8389	92.14	0.439
100	0.7911	0.8488	97.63	0.383
AAL	0.7225	——	29.62	0.332

the priorities stored in the MaxPQ out-of-date. For instance, if components A and B are contracted, and there is a component C with positive links to both A and B, then the $C - A$ and $C - B$ links will be replaced by a $C - (AB)$ link with a different weight. If either $C - A$ or $C - B$ links happened to be the maximum-weighted links of C, then C's priority will be lower, and C ought to be further down the queue.

To address this issue, we could re-compute the priority of every component drawn from the MaxPQ. If the component's actual priority is not the maximum, then it is re-inserted into the queue with updated priority. Additionally, the maximum priority component may no longer exist in the CG due to contraction with another component. In this case it is simply discarded.

Without using an efficient priority queue, the linear searching method of finding the next link to contract results in a $O\big(n(n-k)\big)$ time algorithm. Using the priority queue the time complexity of Edge-Contraction is $O\big((n-k)(m+\log n)\big)$, where m is the average number of positive links a component has.

3.2.2 Results

The Edge-Contract parcellations notably outperformed the anatomical AAL parcellation in the Adjacency-Score. For further comparison, found the mean edge weight in the graph to be 0.7258, which is even slightly higher than the average adjacent within-parcel edge in AAL. This suggests that the AAL parcellation has no connection with the functional information contained in this fMRI data set. It shows on the other hand that the Edge-Contract algorithm can successfully locate regions of functional similarity.

The one apparent deficiency of Edge-Contract is the jaggedness of its parcels. Comparison of our parcellations with the AAL shows that our 116-component parcellation – the same number of components as AAL – has an average parcel surface area roughly $\left(\frac{92.14}{29.62}\right)^{\frac{2}{3}} \approx 2.11$ times that of AAL. Visually, that difference is shown in the plots below of a typical component from each parcellation.

3.3 The Generalized Edge-Contraction Algorithm

In the original Edge-Contraction algorithm, the criteria for selecting the next link to contract was to search through the set of smallest components and find the link of maximal weight. Because this criteria takes no account of the shape of the two components to be contracted, the resulting parcels tend to be very jagged.

To address this we expanded the criterion for finding the next link to contract. Rather than use only the size of the component and the weight of the link, a *Generalized Edge-Contraction* algorithm may use any piece of information stored in the Contractible Graph about a pair of components, such as the number of edges connecting two components. A *priority function* takes information of any two components in a CG and outputs a real number, the priority. For each iteration, the pair of components with the largest priority is contracted and the priorities of neighboring components with respect to the newly conjoined component are computed.

For two components A, B let $|A|$ denote size (number of vertices) of A, $E_{A,B}$ denote the set of edges between A and B, and $w_{A,B}$ the weight of the link connecting A and B. The priority function of the original Edge-Contraction algorithm is $p_0(A, B) =$

Table 3.2: Results of Generalized Edge-Contract for Various Parameter Settings

α	β	Adjacent	Jaggedness	Balance
3	1	0.7500	25.3	0.285
6	1	0.7574	31.0	0.231
10	1	0.7634	36.6	0.211
6	2	0.7565	31.0	0.199
6	4	0.7576	30.9	0.270

(a) 116 Parcels

α	β	Adjacent	Jaggedness	Balance
3	1	0.7592	23.8	0.218
6	1	0.7651	28.0	0.143
10	1	0.7705	32.0	0.221
6	2	0.7670	28.2	0.169
6	4	0.7669	28.8	0.293

(b) 300 Parcels

$w_{A,B} - |B|$.

A link (A, B) will have high priority if either component is small, if the link has a large weight, and if it has a good boundary-ratio, defined as $\frac{|E_{A,B}|}{\min(|A|,|B|)}$, which helps to minimize jaggedness. From these notions we created a family of priority functions indexed by tunable parameters α and β

$$p_1(A, B) = \frac{w_{A,B}^{\alpha}}{|A|^{\beta}} \cdot \frac{|E_{A,B}|}{\min(|A|, |B|)}$$

that modulate the balance of small size, large weight, and high boundary ratio. Table 2.2 shows the results of the 116-component and 300-component Generalized Edge-Contract parcellation when performed for various values of α and β. The adjacent scores have fallen on average from 0.79 in the vanilla edge contraact to 0.75 here. This implies a tradeoff between prioritizing edge weight, jaggedness, and balance in producing a parcellation.

With a balance score of 0.332, the AAL parcellation was better balanced than all of the parcellations obtained via this priority function, though not drastically. In terms of jaggedness, these parcellations all fell within the vicinity of AAL. In this one brain, the best parcellation arises from parameters $\alpha = 6$ and $\beta = 4$.

Parcellations of other brains using generalized edge contract can be found in chapter 8.

Chapter 4

Results

Chapter 5

Conclusion