FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Data-Driven Anatomical Layouting of Brain Network Graphs

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Gwendolyn Rippberger
Matrikelnummer 01307685

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Math. Dr. Katja Bühler
Dipl.-Ing. Florian Ganglberger

Wien, 10. April 2019

Gwendolyn Rippberger	Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Data-Driven Anatomical Layouting of Brain Network Graphs

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Gwendolyn Rippberger
Registration Number 01307685

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Math. Dr. Katja Bühler
Dipl.-Ing. Florian Ganglberger

Vienna, 10th April, 2019

Gwendolyn Rippberger | Eduard Gröller

Erklärung zur Verfassung der Arbeit

Gwendolyn Rippberger
Schelleingasse 6/1, 1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. April 2019

Gwendolyn Rippberger

Acknowledgements

First of all, I want to thank Dipl.-Math. Dr. Katja Bühler and Dipl.-Ing. Florian Ganglberger for all the support and interesting discussions. They showed me what a project looks like in the "real world". A special thanks to Florian Ganglberger for helping me through my technical difficulties during this project.

Futhermore, I sincerely thank my boyfriend for supporting me through the tough times I had during this semester and always being there for me when I needed help.

Abstract

The visualization of brain networks today offers a variety of different tools and approaches. Representations in 2D such as connectograms, connectivity matrices, and node-link diagrams are common but an abstract visualization of the network without any anatomical context. Visualizations tools show anatomical context in 2D but adjust it especially for a certain species as for example the fruit fly's brain. This project presents a tool for data-driven brain network visualization using the open-source graph library Cytoscape.js to avoid hard coded spatial constraints. The goal of the project was to find a layout algorithm that resembles the anatomical structure of the brain visualized without any hard coded constraints. After testing the layouts, they have been evaluated on different properties like symmetry, node overlapping, and anatomical resemblence. Additionally, we conducted an open discussion with collaborators of the Research Institute of Molecular Pathology (IMP) in Vienna and present the results.

Contents

CHAPTER 1

Introduction

The brain's complex structure and how it organizes itself has always been one of the main areas of interest in neuroscience. With today's technical possibilities to get brain network data, there are a variety of tools available that focus on different aspects of the brain as well as different species and different dimensions. Brain network visualizations have different approaches depending on the focus of the project and the data being visualized. Those tools are mainly developed to help neuroscientists to explore brain data and to develop hypotheses but also to visualize for explanation purposes.

For 3D data of brain networks including brain regions and their network, the most intuitive way is to visualize it in 3D. A common way is to visualize it as a 3D node-link network with edges rendered as straight lines and nodes rendered as spheres [LDTS14, XWH13]. Edges can also be rendered as curved lines along the brain surfaces [LFG+15]. However, visualizing a complex brain network with a large number of connections in 3D can have many obstacles. Cluttering and occlusion can be minimized by giving the user the possibility to filter and select. Still, a chosen selection of interest may have a high number of connections which can still be difficult to analyze in 3D because of the depth and possible occlusion. Taking a brain's complexity, it is difficult to show the highest level of detail and still see the entire global structure. Different methods like level-of-detail-visualization [BD07] or edge bundling [BSL+14] can be used to represent the hierarchical level. User can be given a reference through linked views of the brain network and its anatomical context [MBB+17] working together with color coding the nodes with the color of their region (according to the Allen Brain Atlas, etc.).

A different approach is to visualize the network without any anatomical context as node-link diagrams [nod], connectivity matrices [MM14] or connectograms [IDVH14]. Linked views give the flexibility to choose different 2D representations of the network and still have the anatomical view at the same time. Working with node-link diagrams,

it is difficult to visualize a brain network giving a context with which the user can easily understand the structure of the layout and find connections. Spring-embedded layouts usually provide good results for node-link diagrams regarding node distribution and overlapping. Most of them also support network properties to adjust the layout, so that, for example, well-connected nodes are closer and insights of the network connections can be gained by just taking a look at the layout [APLK+15].

NeuroMap [Sor13] presents an anatomical layout using 2D spatial constrains emulating an abstract view of the fruit fly's brain. Together with 3D visualizations, showing the neurons in their anatomical context and rendering potential neural connections as interactive circuit-style wiring graph, this tool was made to explore data and support hypothesis formation. However, this solution was adjusted to visualize a drosophila brain (fruit fly) and would need to be reprogrammed to show any other species.

The goal of this thesis is to develop a tool with the open-source graph library Cytoscape.js that uses a data structure containing anatomical and structural connectivity information about the brain and layouts it data-driven according to the given data in a 2D node-link network. In addition the layout should have the following features:

- reflecting the layout of the brain anatomy giving anatomical context
- should be data-driven, (in the best case) no hard coded spatial constraints
- should be reproducible as the same data should result in a similar layout (to a certain extend)
- should be visually traceable as the layout should not change too much regarding changes in the hierarchie
- nodes should in the best case not overlap or the overlap should be minimized

The data-driven layout should solve the process of having to hard code spatial constraints regarding the species being visualized. This tool should then result in a graph that reflects the anatomy of the visualized brain. Regions rendered as nodes that are anatomically close should then also be close in the layout. After this tool is sophisticated enough it should be integrated into the application for brain network visualization [GSF+18].

CHAPTER 2

Methodology

Firstly, the data and the data structure used will be presented. Then the most promising candidates of the layout algorithm supported by Cytoscape.js will be introduced. Afterwards the main concept of how the layouting process with the anatomical connectivity works and the visualization concept used in this work are presented.

2.1 Data

The data visualized in this project is provided by different brain initiatives as the Allen Institute [all] or the Human Brain Project [brab]. It includes a hierarchical definition of the brain regions and connectivity data. The connectivity data that was used can be divided into two main types:

- *Anatomical Connectivity* describing neighbouring regions, so only regions that are neighbours have edges. The edge weight is calculated by how voxels in a 132x80x114 mouse brain are structurally connected. Larger touching areas result in a higher edge weight. To regulate the case that large brain regions also have larger surfaces the *Normalized Anatomical Connectivity* divides the value by the size of the region.

- *Structural Connectivity*: with the voxel-wise structural connectivity matrix from Ganglberger et al. [GKP+18] we have information about how voxels in a 132x80x114 mouse brain are structurally connected. The edges directed and their weights are normalized to a range from 0 to 1.

The structural connectivity was chosen to be visualized in this work but there are other possible networks like functional connectivity or spatial gene expression correlation networks.

2.2 Data Structure

The data structure developed by Ganglberger et al. [GKHB18] manages to aggregate queries in real-time. Aggregated queries describing aggregated connectivity from, to, or between brain areas. It includes hierarchies of brain parcellations but only on the left and right but not on the front and back and the up and down.

2.3 Layout Algorithms

2.3.1 Compound Spring Embedder

The Compound Spring Embedder (CoSE) is a physics simulation layout. U. Dogrusoz et al. [DGC+09] first introduced a new algorithm for the layout of undirected compound graphs dealing with multi-level nesting, links to non-leaf nodes in the nesting hierarchy, and inter-graph edges that may span multiple levels of nesting. Furthermore, their algorithm is able to handle non-uniform node sizes.

Force-directed layout algorithm [FR91, EAD84] work with pulling and repelling forces. Nodes simulate the physics of electrons and edges those of springs till they come to an equilibrium state. Repulsion acts on objects that are "too close" to avoid nodes from overlapping and an additional gravitational force is used to keep the graph components together. Edges work as springs that attract adjacent nodes. Their modified spring embedder system includes compound nodes and inter-graph edges in the physical system.

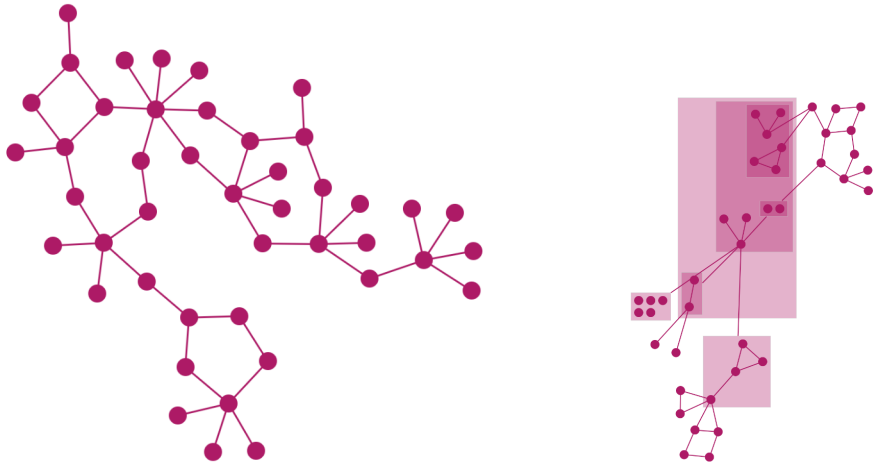
Cytoscape offers two different implementations of this algorithm, which for distinction purposes are going to be mentioned as "CoSE" and "CoSE-Bilkent".

CoSE. This layout based on the algorithm of U. Dogrusoz et al. [DGC+09] was implemented by Gerardo Huck as part of Google Summer of Code 2013 (Mentors: Max Franz, Christian Lopes, Anders Riutta, Ugur Dogrusoz) [cosb]. It works well on noncompound graphs and has additional logic for compound graphs. It is less computational expensive than the CoSE-Bilkent, still producing good results. Layout algorithms can be tweaked in cytoscape with the properties provided. Choosing the node repulsion multiplier, ideal edge length, and edge elasticity the results can be optimized.

CoSE-Bilkent. Taking the CoSE implementation, it performs poorly (regarding quality) due to assumptions and simplifications, especially with compound graphs. The CoSE layout by Bilkent [cosa] provides a solid implementation of the presented algorithm [DGC+09]. Although, it is computational more expensive the results are better (taking criteria as symmetry and node placement). The demo presented by Cytoscape results in layout that can be seen in Fig. 2.1

2.3.2 Constraint Layout

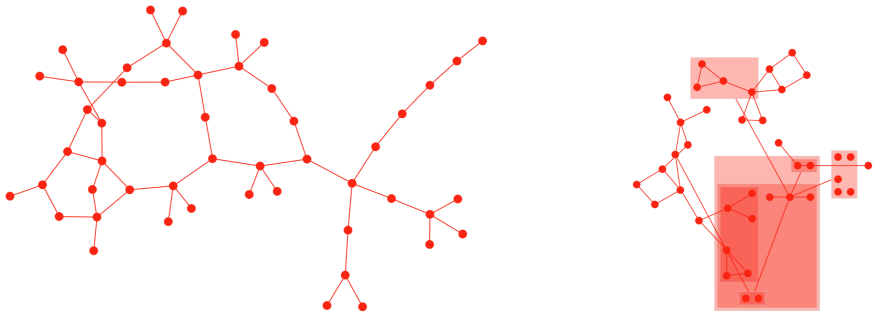
The constraint layout (cola) [col] is based on a complete rewrite in Javascript of the C++ libcola library. Introduced by Dywer et al. [DKM06], this algorithm allows separation



(a) (b)

Figure 2.1: Cytoscape.js Demo using the Cose-Bilkent Layout with 2.1b and without 2.1a compound nodes [cosa].

constraints to enforce a minimum separation between a pair of nodes. Overlap avoidance is a built-in feature of Cola. Unless particular options are specified that make overlap avoidance impossible, Cola will in generally produce results where the nodes do not overlap. Results of this layout algorithm can be seen in Fig. 2.2.



(a) (b)

Figure 2.2: Cytoscape.js Demo using the Cola Layout with 2.2b and without 2.2a compound nodes [col].

2.3.3 Spread

The spread algorithm [spr] focuses mainly on spreading the nodes so that the whole view port is used. It consists of two phases: the positioning of the nodes through a chosen layout algorithm and afterwards spreading the nodes evenly. The implementation provided in Cytoscape.js uses the Fruchterman-Reingold algorithm [FR91] initially (using

the CoSE presented in the section 2.3.1) and then Gansner and North algorithm [GN98] to spread the nodes over the available space.

In the first phase by default the embedded CoSE algorithm is used because it works fast. Alternatively, other algorithms can be chosen or the first phase can be skipped and the node's existing positions are used for the next phase.
In the second phase the nodes are spread using Steven J. Fortune's algorithm [For86] to create Voronoi diagrams. As you can see in Fig. 2.3 this layout results in a good distribution of nodes using all space available.

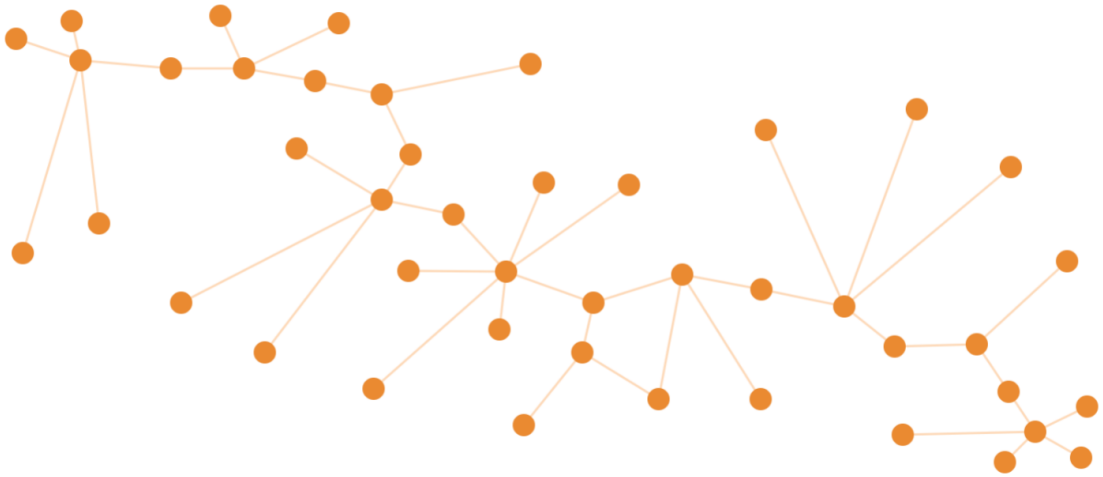


Figure 2.3: The Cytoscape.js Demo using the spread layout algorithm provided [spr].

2.3.4 Klay

Based on the paper from Schulze et al. [SSvH14] their algorithm (which was implemented into cytoscape) was originally developed for data flow diagrams. Ports describing dedicated connections points with which nodes are connected. Port constraints define where ports are being place on the corresponding node. Originally this approach was developed for layered graphs (introduced by Sugiyama et al. [STT81]) which draw vertices of a directed graph in layers (either as rows or as columns shown in Fig. 2.4).

2.4 Anatomical Layouting

The concept of the data-driven layout is to use the anatomical connectivity mentioned in section 2.1 and use it for the layout process of the graph. After the layouting process is done, the edges are removed and replaced by the edges in our case representing the structural connectivity because the structural connectivity is what neuroscientists want visualized. It contains the information about the brain network. Nevertheless, any connectivity mentioned in section 2.1 can be visualized.

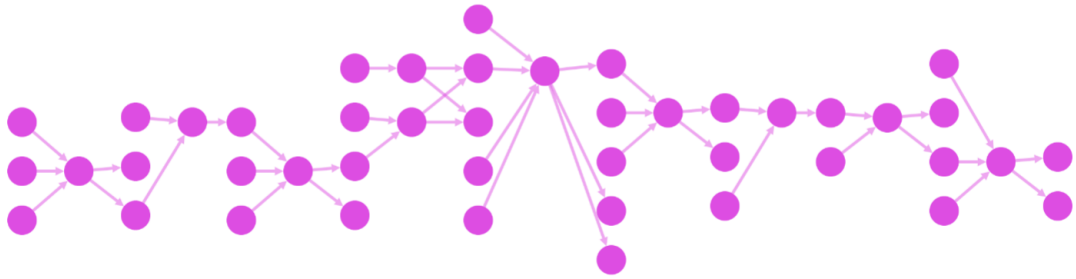


Figure 2.4: The Cytoscape.js Demo using the klay layout algorithm [kla] provided.

To enable a smooth transitioning because this tool was developed for a web application where user can switch in between different hierarchy levels we differentiate between two different layout states: the layout initialization phase and the updating. The layout is initialized with random node positions. For updating, the precalculated node positions are taken and the incremental mode is supported.

2.5 Visualization Concepts

The brain regions rendered as nodes are colored with the regions colors using the Allen Brain Atlas color scheme. Additionally, arrows are rendered to show the direction of the connection between nodes. The strength of the connection is mapped to the opacity of the edge and the hue of the arrows (seen in Fig. 2.5). Nodes are labelled with the short name of the region they represent.

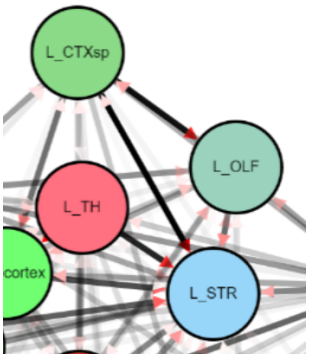


Figure 2.5: Close up of a graph

CHAPTER 3

Implementation

In this chapter the technologies that were used during this project are briefly explained following implementation details. Additionally, two different layout approaches are presented using hierarchical information and the compound node support. As the results did not show any symmetry or anatomical resemblence, the approaches were not further developed.

3.1 Technologies

3.1.1 React

The React [rea] interface library offers a component-based way to create interactive user interfaces. Instead of loading entire pages React only updates and renders the component where data changes. This requires that each component is independent and manages their own state. Components have the possibility to be stateless and just work with input data. In our case the user selects different hierarchies and brain regions, which are saved in a state passed on to the React component. Components can also be stateful, controlling their internal state data, which is private.

3.1.2 Redux

Working with different views and user interactions Redux [red] provides a centralized state container for Javascript apps. Redux works with a single *store* that saves the state of the app as object tree. This store can only be changed by emitting *actions* that describe what happened. *Reducer* then specify how the action modifies the state of the app. Working with Redux makes the application predictable with consistent behaviour and easy to debug. In our case when a selection is made by the user, the selection is saved in the store and then passed on to all the other components, including the 2D graph visualization.

3.1.3 Cytoscape.js

Cytoscape.js [cyt] is an open-source graph theory library fully written in Javascript. It is used for graph analysis and visualization. The library was created at the Donnelly Centre at the University of Toronto and is the successor of Cytoscape Web.
It supports user interaction with all gestures out-of-the-box as pinch-to-zoom, panning, drag-and-drop, etc. and can be easily integrated using package managers like npm, yarn or bower. Instances of Cytoscape.js correspond to a graph. It offers a range of different possibilities to manipulate the graph's properties as for example layout, animation, and interaction. Besides basic layout algorithms that are embedded in the package such as grid, circle, concentric, etc. it offers more complex layout algorithms as extensions which were mainly used during this project.

3.2 React-2D-Network

The current prototype of the tool was programmed as a React Component that can easily be integrated into the framework once it is sophisticated. It was developed, tested and executed on a laptop using NVIDIA GeForce GTX 1050Ti GPU, an Intel Core i7-7700HQ 2.8GHz CPU, 8 GB RAM and a 128 GB SSD.

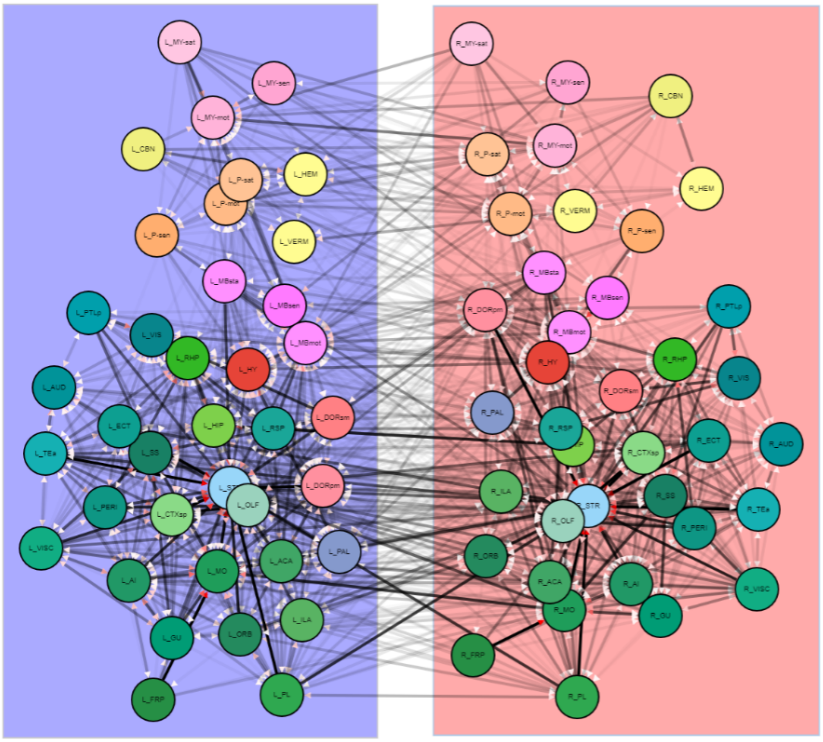
The data structure presented in 2.2 is passed to the graph's React component and consists of the user's selection. The selection includes the structural, anatomical, and normalized anatomical connectivity and the brain regions selected. As all connectivities are passed as one JSON object, it is filtered and split into the three different types. The "cm" property of the JSON file represents which type of connectivity is represented with #121:0 being the structural connectivity, #122:0 the anatomical connectivity and #123:0 the normalized connectivity. The regions are passed separatly.
After those have been saved the regions are mapped from the props by using "this.props" and for each region a node object according to the Cytoscape.js constraints. Then the edges are created in a similar process. An additional counter for each edge was added because per graph each graph object must have a unique id.

To modify a graph there must be a reference to the Cytoscape object. The react-cytoscapejs package that was used in this tool offers a cy prop allowing to get a reference Cytoscape object. With the cytoscape object, which represents the graph, nodes, and edges can be retrieved for updating manually. To have a smooth transition between layouts that support an incremental mode, node positions must be saved. For this the old nodes from the graph are retrieved and compared with the current selection. Nodes that are not in the current selection are removed and nodes which are new are added each one at a time. Selections cannot be updated with the `cy.add(nodes)` command as nodes with the same id are overriden and the position gets lost.

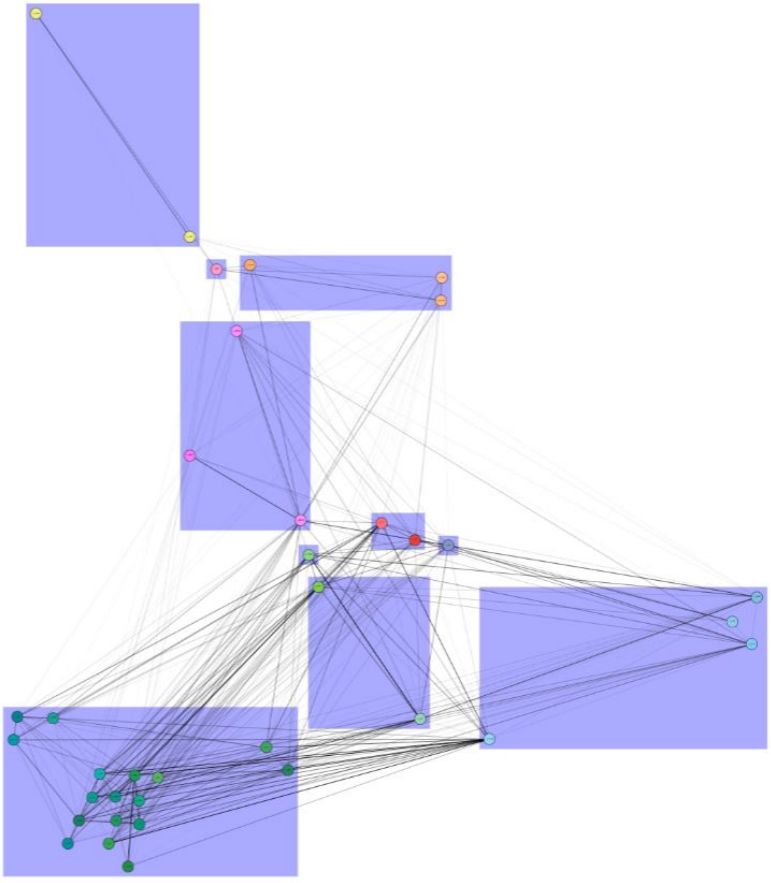
For performance reason we set a threshold to render only edges that have a weight

above 0.1. Then for the final cytoscape component we set the layout to 'preset', which takes the nodes' precalculated positions for rendering.

As most of algorithms support compound nodes, we tested two possible use cases: hierarchical layout (brain regions that have the same parent node are rendered as compound nodes) and anatomical layout (parent nodes being the left and right side of the brain). For this an additional JSON file is needed containing the whole hierarchy of the brain, which is then used to save each node with its according parent as a cytoscape node object. Then the graph is rendered as described before. This approach results in a layout seen in Fig. 3.1b. The brain hierarchy is unbalanced, which results inan unbalanced ratio of the compound nodes. The compound nodes scenario was tested using the CoSE layout algorithm.
Working with the anatomical layout, two nodes were added reflecting the left and right part of the brain (the left side being represented with a blue background and the right side with a red background seen in Fig. 3.1a). Working with the CoSE-Bilkent because it has better results regarding symmetry, this approach reaches a similar output to not using compound nodes but nodes a clearer separated using parent nodes.



(a) Compound nodes, splitting the nodes into the left and right side of the brain using CoSE layout algorithm



(b) Nodes with the same parent rendered as a compound node using the CoSE layout algorithm

Figure 3.1: Compound and hierarchical layout result

CHAPTER 4

Evaluation

After implementing the different layout algorithms offered by Cytoscape.js, the graphs were evaluated regarding symmetry, anatomical layout, reproducibility, and visual tracability. The perfomance was also measured as it is important that the layouter runs in real-time in a web-application. A user study was conducted to gain a general impression on how the graph is perceived (regarding the properties like symmetry, node overlapping, etc.) and a quantitative comparison of the symmetry and performance. We evaluate the graph layout algorithms taking fixed selections of brain regions and (taking the most promising layout algorithm) we tested two different use cases on how it works regarding the layout's stability. The first a selection that chooses regions deep in the hierarchy and the second use case tested a selection opening and closing the left side and the right side respectively.

4.1 Quantitative Evaluation

4.1.1 Measures and Test Cases

To evaluate the layout algorithms regarding performance and appearance, we decided on four different selections that should cover basic user selections:

- only the left side of the brain with 36 regions and 182 anatomical connections, hierachy levels seen in the Fig. A.9a
- a small selection of the brain with 34 regions and 211 anatomical connections
- a average selection of 76 regions and 519 anatomical connections
- a bigger selection including 100 regions and 688 anatomical connections

The tool should later be integrated into a framework that works with a parcellation browser showing the hierachy level of the brain regions selected. For this for each selection the selection in the parcellation browser is shown in A.9. To compare symmetry for each layout a line was drawn by myself which should reflect the axis that separates the left from the right part of the brain. The line should split the nodes into two equally sized groups. Then the number of wrongly placed nodes (left nodes on the right side and right nodes on the left side) is determined. As the counting was done manually only the 34 regions selection was taken for comparison. Due to the biological evolution most brains in general are symmetrical regarding the left and the right side. So the amount of wrongly placed nodes additionally indicates anatomic resemblence besides the symmetry. Considering this, the wrongly placed nodes were counted of the left and right side of the brain to indicate anatomic resemblence and symmetry. However, layouts also showed no symmetry axis and were classified as not symmetric.

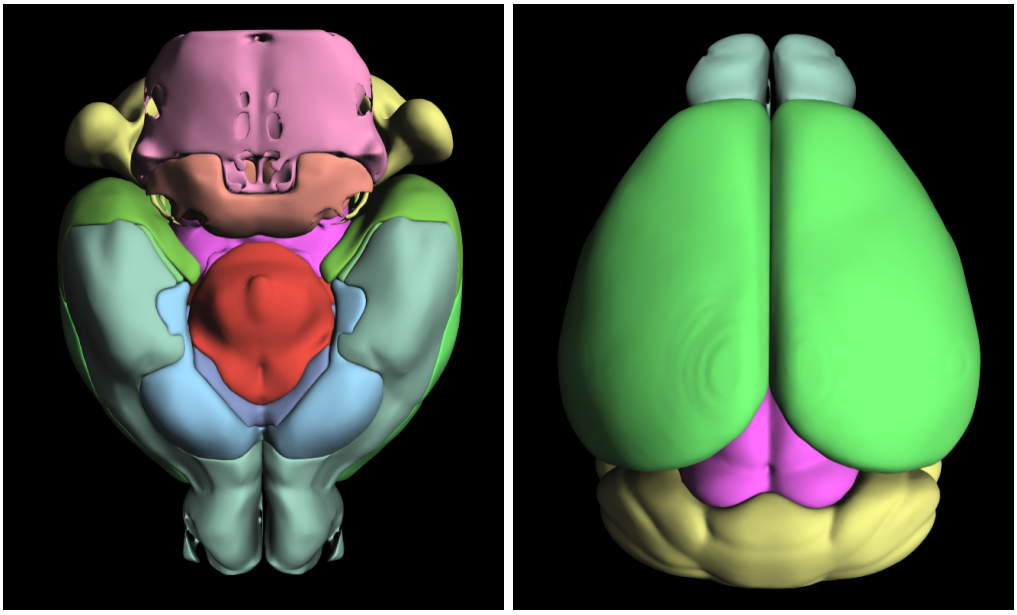
Regarding performance we used the normalized connectivity as the edge weights have no impact on the layout but the connections. Because the Klay algorithm was originally intended for directed graphs and does not work for undirected graphs, it was tested with the structural connectivity (which is directed) and the results were still compared with the other layouts. It is only measured how long the layouting algorithm takes to stop.

To show if the layout reflects the anatomy of the brain the coronal and sagittal views of the adult mouse brain taken from [braa] were chosen to compare with the layout. As the 2D View only has the coronal and the sagittal view, the 3D View was additionally considered to have a overall view of the brain for the comparison.

After comparing the layout properties as mentioned above (results A.1) we took the CoSE-Bilkent because it was the most suitable layout algorithm regarding symmetry and anatomical layout. Then it was tested with two different use cases how smooth the layout transitions between different selections with different hierarchy levels.
The first use case takes the 34 regions selection and navigates through different hierarchy levels with the last one being the one with the highest resolution level and then closing the selection. The second use case one we test how the layout reacts regarding navigation through the hierarchy. Once opening the left side and then the right side and then following by closing each one at a time.

4.1.2 Results

Layout
The CoSE algorithm is a generally good layout algorithm regarding node distribution taken for example Fig. A.1. Because of the randomized start positions of the nodes there is no reproducibility of the results. Nodes do not overlap but there is no visible symmetry. Nevertheless, the layouts sometimes result in an anatomical partition of front and back (as seen in Fig. A.1c). Taking the node positions usually the nodes that are



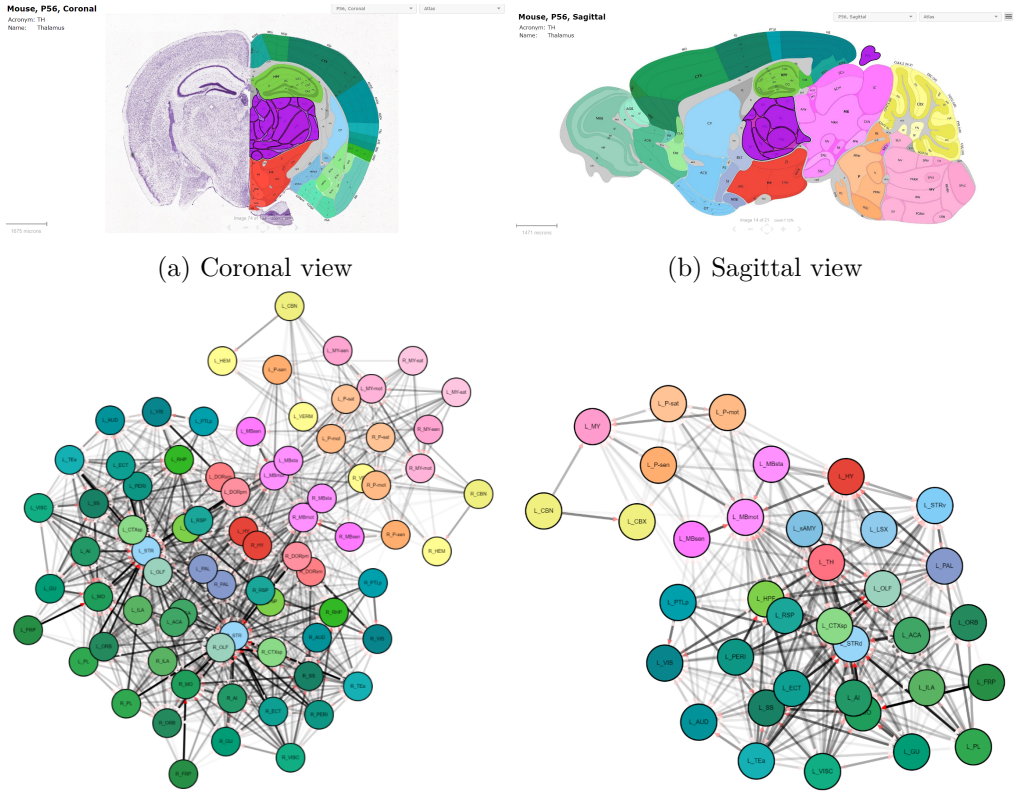
(a) View from underneath in 3D (b) View from above in 3D

Figure 4.1: 3D views of the mouse brain taken from [braa]

anatomically close are afterwards within a reasonable distance.

The CoSE-Bilkent algorithm outperforms the other layout algorithms regarding all properties see Fig. A.2. For symmetrical selections (same nodes chosen from the right and left side of the brain), all test cases result in a visible symmetry. Also regarding the anatomical reflection (as seen in the underneath view in Fig. 4.1a and for the layout result in Fig. 4.2c front nodes and back nodes are clearly recognizable. Comparing only the left brain side selection with the sagittal view seen in Fig. 4.2b, symmetry cannot be measured as the right side is missing but there is still an anatomical layout visible. Front and back part can be separated and the overlaying isocortex (green) includes the striatum (blue) and interbrain (red) as it overlays those brain regions.
Spring embedder algorithms (including the CoSE algorithm presented in section 2.3.1 apply repulsion forces on nodes to avoid overlaps and put some minimal distance between nodes. Overlapping cannot be avoided in our case because our graph is very dense and the algorithm cannot satisfy all constraints. As the the CoSE-Bilkent algorithm provided the best results, we decided to test how smooth it transitions between the selections using the test cases described in section 4.1.1.

The spread algorithm (taken the CoSE-Bilkent algorithm as pre-layout) results in a very wide spread diagram (see Fig. A.3). Through the adjustment of the properties there is no tighter result possible where nodes are positioned closer together. There is no visible symmetry in the layout but there are no overlaps. Node positions cannot be traced back to anatomical context.



(a) Coronal view

(b) Sagittal view

(c) CoSE-Bilkent Result for both brain sides and 76 regions in total

(d) CoSE-Bilkent Result for only the left side of the brain

Figure 4.2: 2D views of the mouse brain [braa] and the 2D network result for the CoSE-Bilkent Layout

The Cola Layout results (see Fig. A.4) in a good node distribution without any overlaps, but without any anatomical reference possible or symmetry visible.
Altough the Klay algorithm does not show any anatomical context or symmetry, it does (partially) reflect the hierarchy of the brain (see Fig. A.5). It can later be used especially for the hierarchy of the brain regions showing their structural connectivity offering a way to structure the flows.
A summary of the layout results can be seen in Table A.1.

Performance
Taking that starting point of the layout process till the moment the component is rendered, most of the algorithms perform similar (seen in Table 4.1.2). So the perfomance does not include preprocessing of the data.

Table 4.1: Time measured includes the post-processing. The Klay algorithm was tested with the structural connectivity as with the anatomical connectivity it does not result in a comparable graph.

	left side	34 regions	76 regions	100 regions
Cose	475.3	448.4	6967.9	15251.5
Cose Bilkent	571.8	543.0	7310.3	16029.7
Spread	549.8	500.8	7293.9	15787.7
Cola	484.4	458.8	6930.1	15095.6
Klay	36168.8ms	21000.1ms	2832.9ms	3958.3ms

4.2 User Study

To take an additional qualitative measure into account we conducted a user study with four collaborators from the Research Institute of Molecular Pathology (IMP) in Vienna. We worked together with two PhD students, one postdoc and a technician. All of them were familiar with the Allen Brain Atlas color scheme of the brain regions and are currently working in the area of brain circuit research. Before starting the user study they were all quickly introduced into the framework this tool should later be integrated, so the workflow in which this tool is suppose to be used is clear.

It was generally tried to get feedback on the CoSE-Bilkent results as they were the most promising ones. Different opinions were gathered and are now represented in this thesis.

At first it was mentioned that the visualization should in general be adjustable for colorblind people. In our case we have additional labels and connectivity strength is represented by opacity and this visualization can be seen by a colorblind person.
Secondly, general orientation in the graph was discussed and what could help improve it. As the layout algorithm tested during this thesis does not yet support a constant position of the node, this was complained. Constant positions meaning that there is no rotation of the graph and nodes stay within the same area. It was mentioned that additional axes could help with the orientation of left/right and front/back. It was said that to give better orientation an either abstract view (such as connectograms and similiar) or a complete sagittal or transversal layout should be shown, everything inbetween can be confusing. At last it was mentioned that outlining regions could help match nodes faster.

CHAPTER 5

Future Work

The experiences collected through this project lead to a lot of ideas and opportunities for improvement. Starting with the layout algorithms the Cytoscape.js library not yet offers a variety of layout algorithms applications such as Cytoscape itself or yEd can be used to visualize the data and after having better candidates those algorithms can be implemented as layout extension or be exported to be visualized with Cytoscape.js.

After trying more layout algorithms the visualization can be improved with the points mentioned in the user study regarding rotation and showing axes to help orientation. The relayouting between selections has yet been only implemented for the CoSE-Bilkent algorithm because it was the one with the best layout results. To make the transitions more visual tracable an animation can be included to make the changing of the node positions better visible.

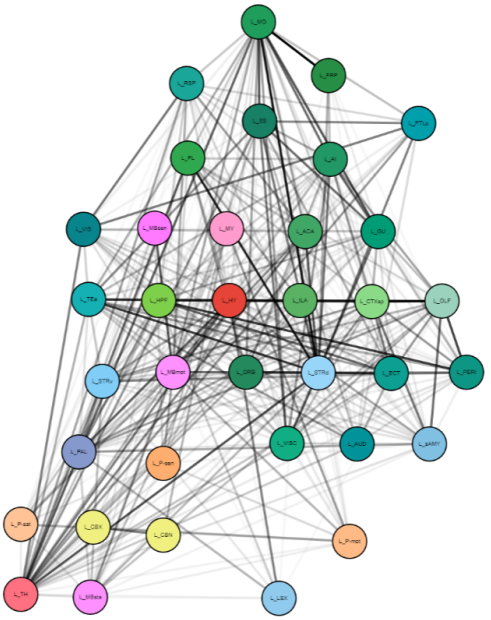
Also the CoSE-Bilkent implementation can be adjusted to perform better for our case. The layout algorithm performance is decided on the implementation but the pre- and post-processing of the data can be speed up with, for example, an extra data structure only saving the difference between selections. That way the difference between the nodes selected must not always be computed and can save time. All the layout algorithms are slow for large graphs with more than 500 edges and 70 nodes. For this case it can be taken into consideration to adjust layouts to our purposes and use heuristics to speed them up.

APPENDIX A

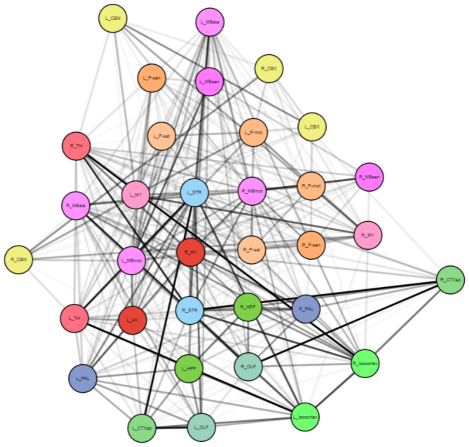
Layout Results

Table A.1: Summary of the layout evaluations

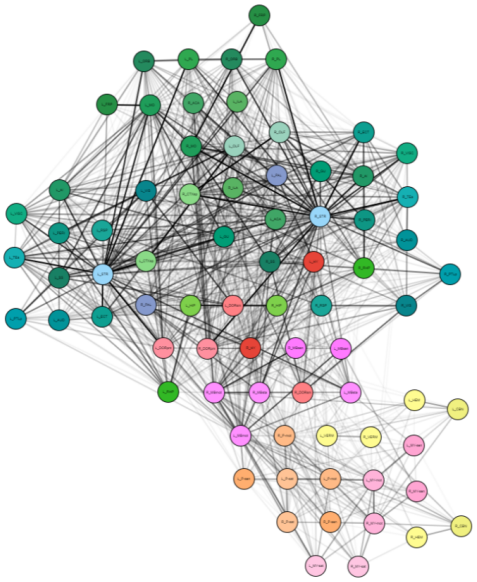
Layout	**CoSE**	**CoSE-Bilkent**	**Cola**	**Spread**	**Klay**
Layout of 34 regions selection	A.1b	A.2b	A.4b	A.3b	A.5b
Visible Symmetry	No	Yes	Yes	No	No
Visible Front and Back Separation	Sometimes	Yes	No	No	No
Visible Left and Right Separation	No	Yes	No	No	No
Node Overlapping	No	Yes	No	33	No
Total Amount of Displaces Nodes (Left and Right)	9	0	16	12	-
Relative Amount of Displaced Nodes	26.4%	0.0%	47.0%	35.2%	-



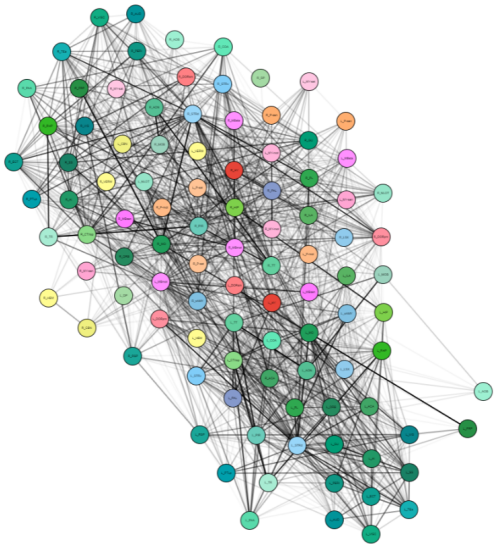
(a) left brain selection



(b) 34 regions

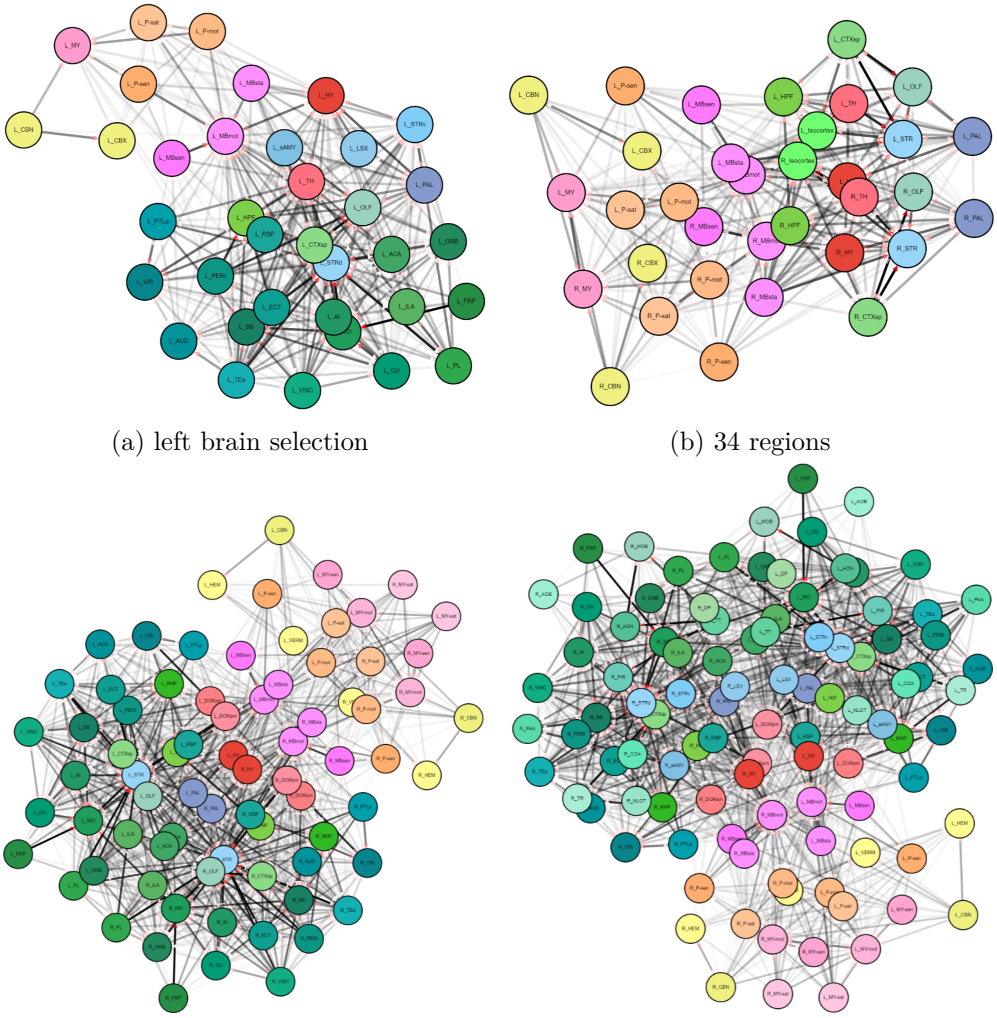


(c) 76 regions



(d) 100 regions

Figure A.1: CoSE Layout Results



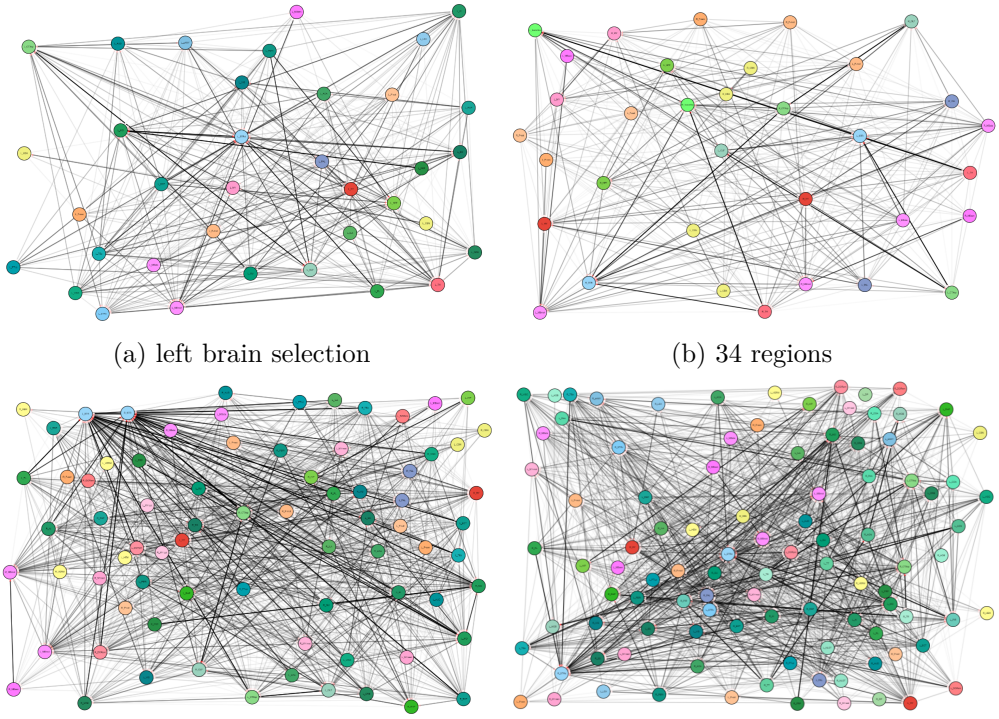
(a) left brain selection

(b) 34 regions

(c) 76 regions

(d) 100 regions

Figure A.2: CoSE-Bilkent Layout Results



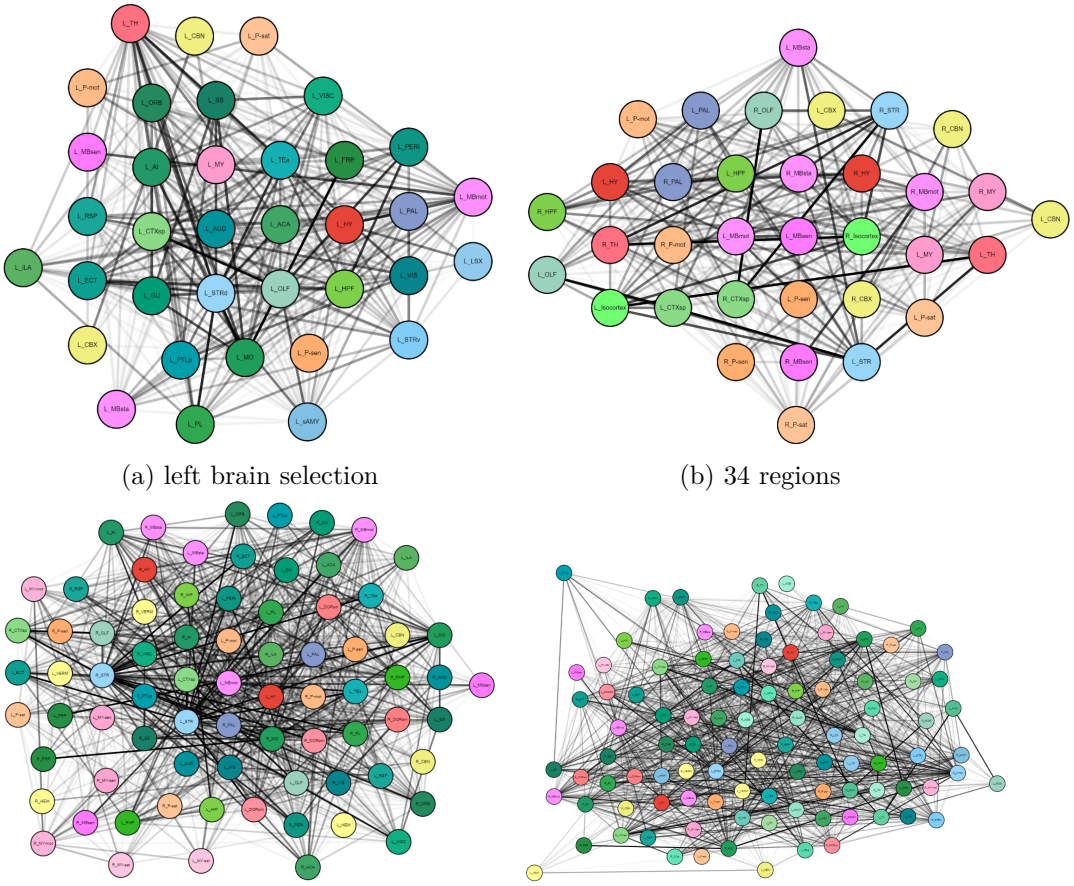
(a) left brain selection

(b) 34 regions

(c) 76 regions

(d) 100 regions

Figure A.3: Spread Layout Results



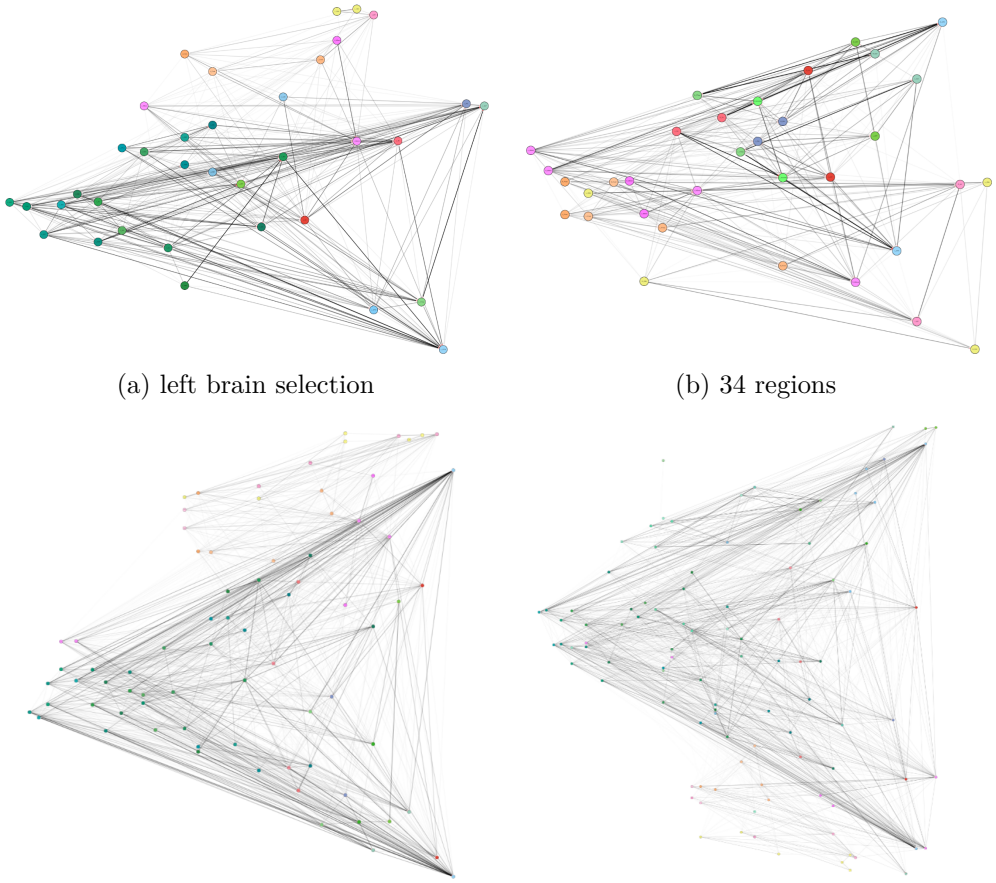
(a) left brain selection

(b) 34 regions

(c) 76 regions

(d) 100 regions

Figure A.4: Cola Layout Results



(a) left brain selection

(b) 34 regions

(c) 76 regions

(d) 100 regions

Figure A.5: Klay Layout Results

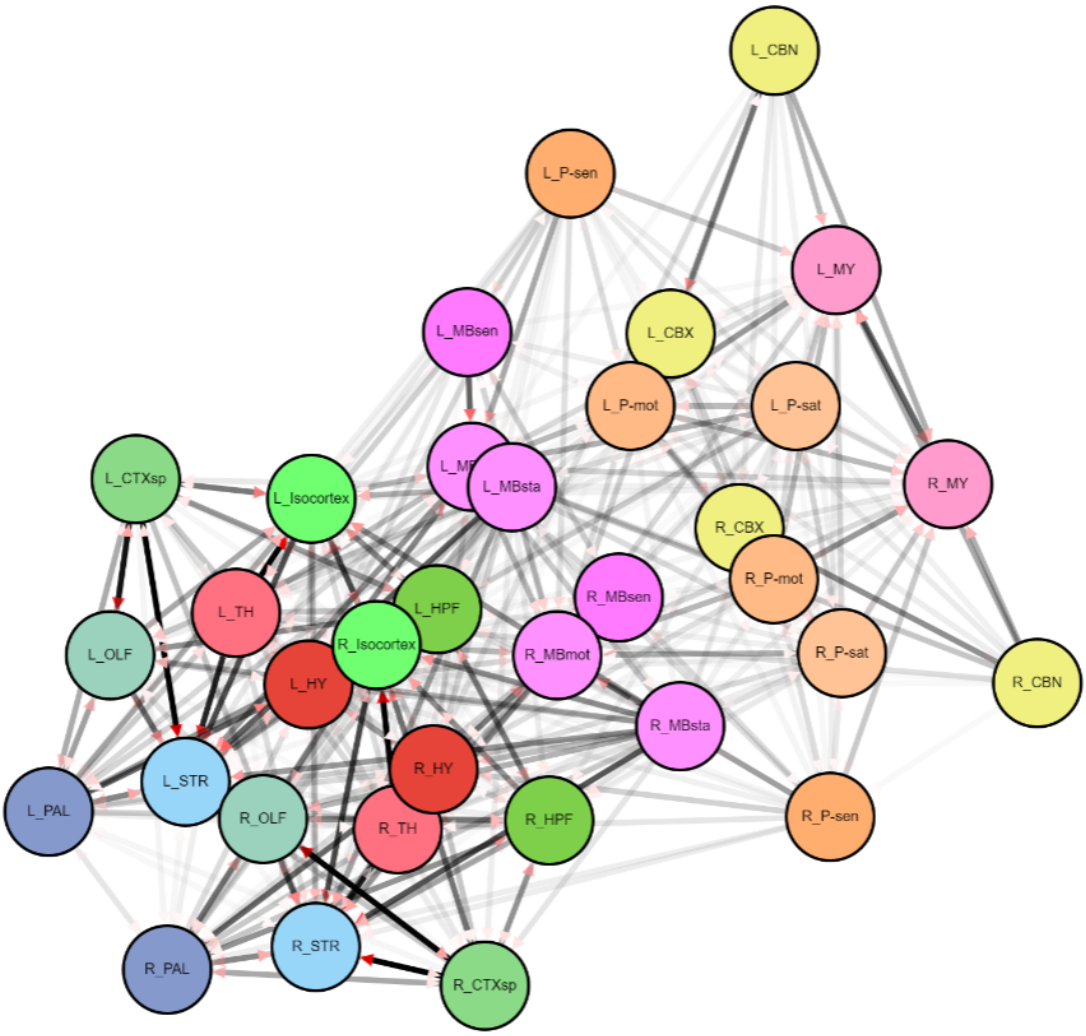
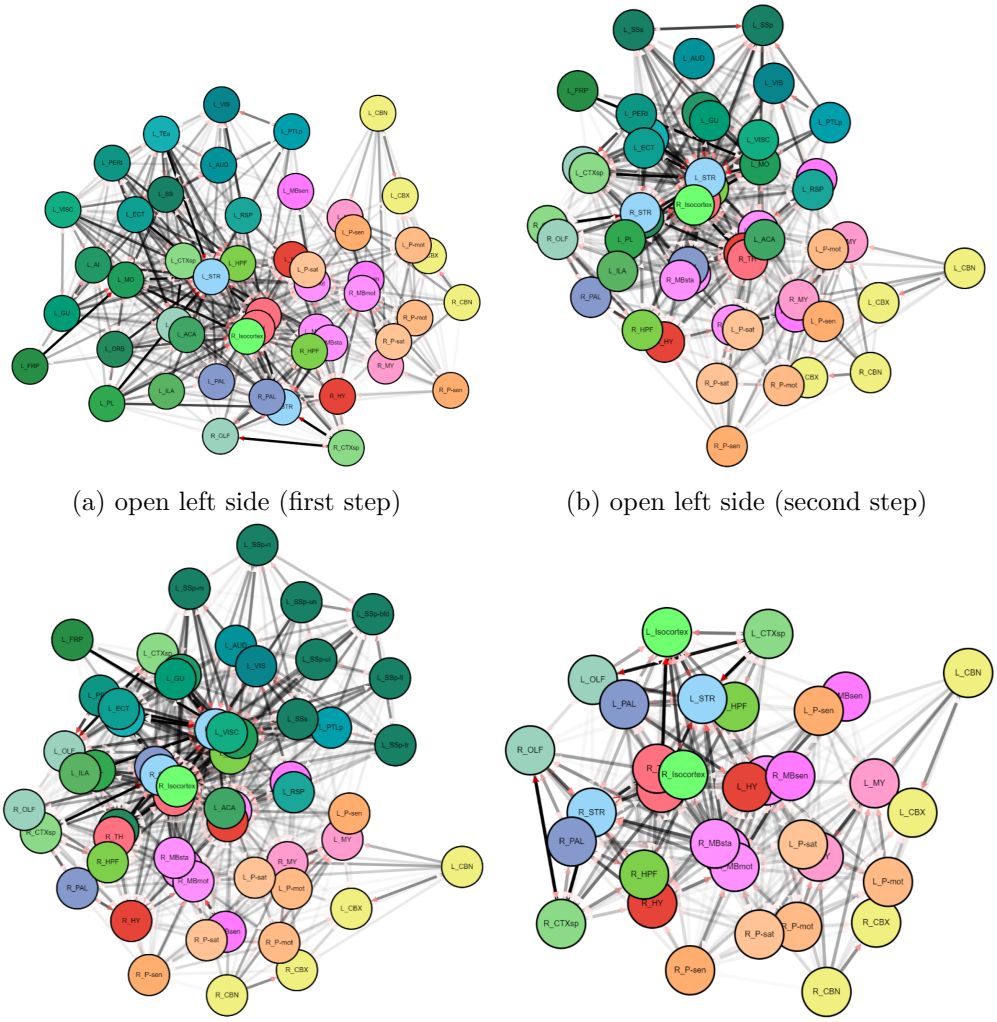


Figure A.6: The initial state for both dynamic test cases



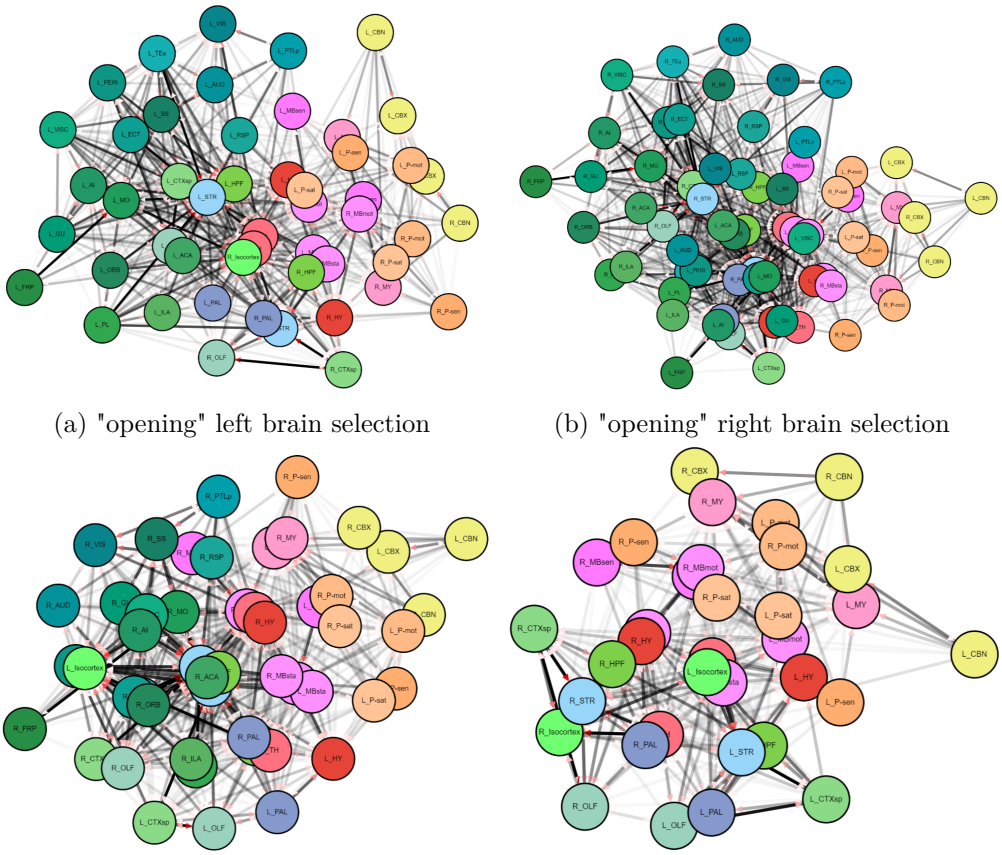
(a) open left side (first step)

(b) open left side (second step)

(c) open left side (third step)

(d) Closing left side selection

Figure A.7: Deep hierarchy selection layout results



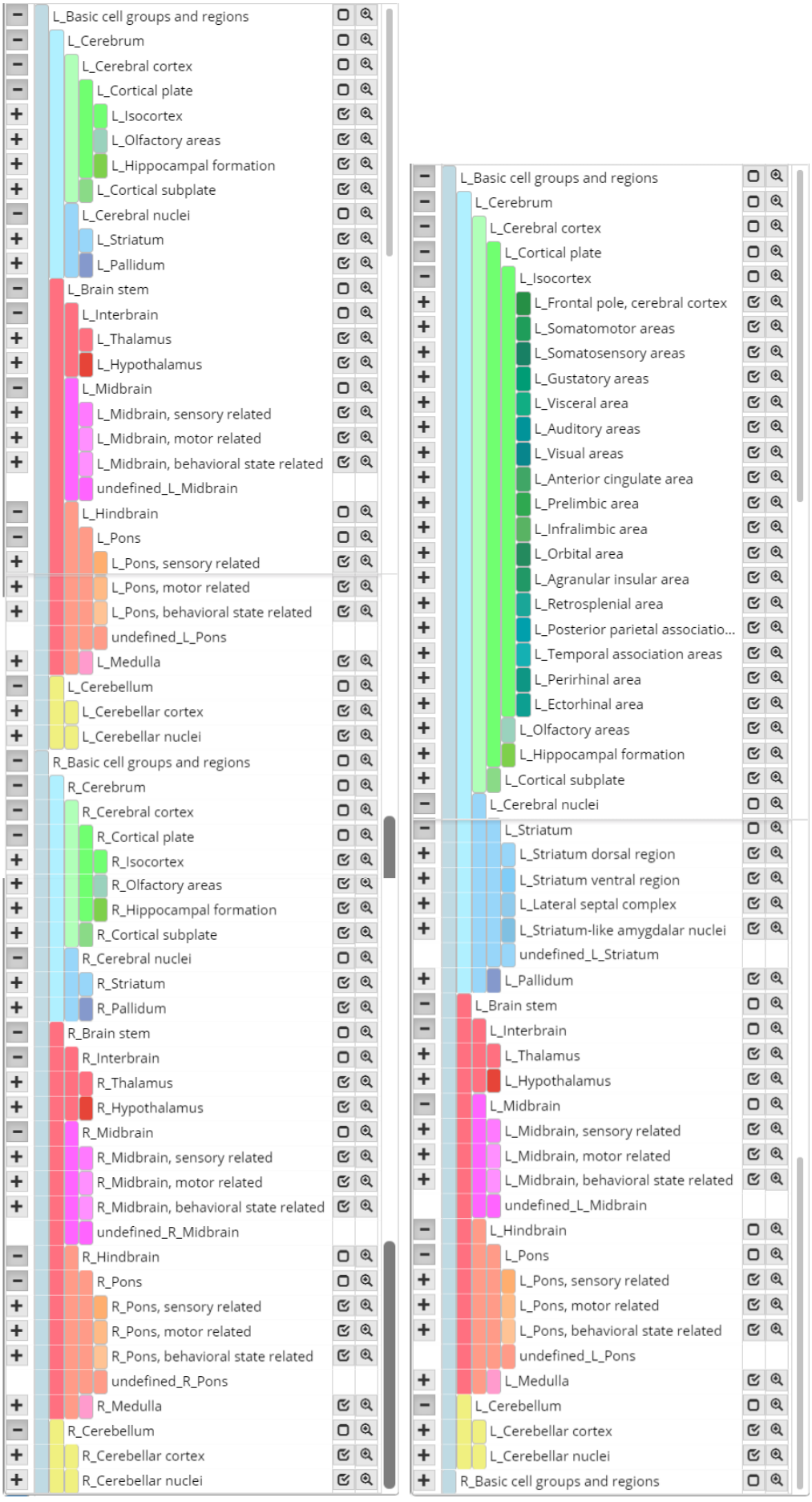
(a) "opening" left brain selection

(b) "opening" right brain selection

(c) "closing" left brain selection

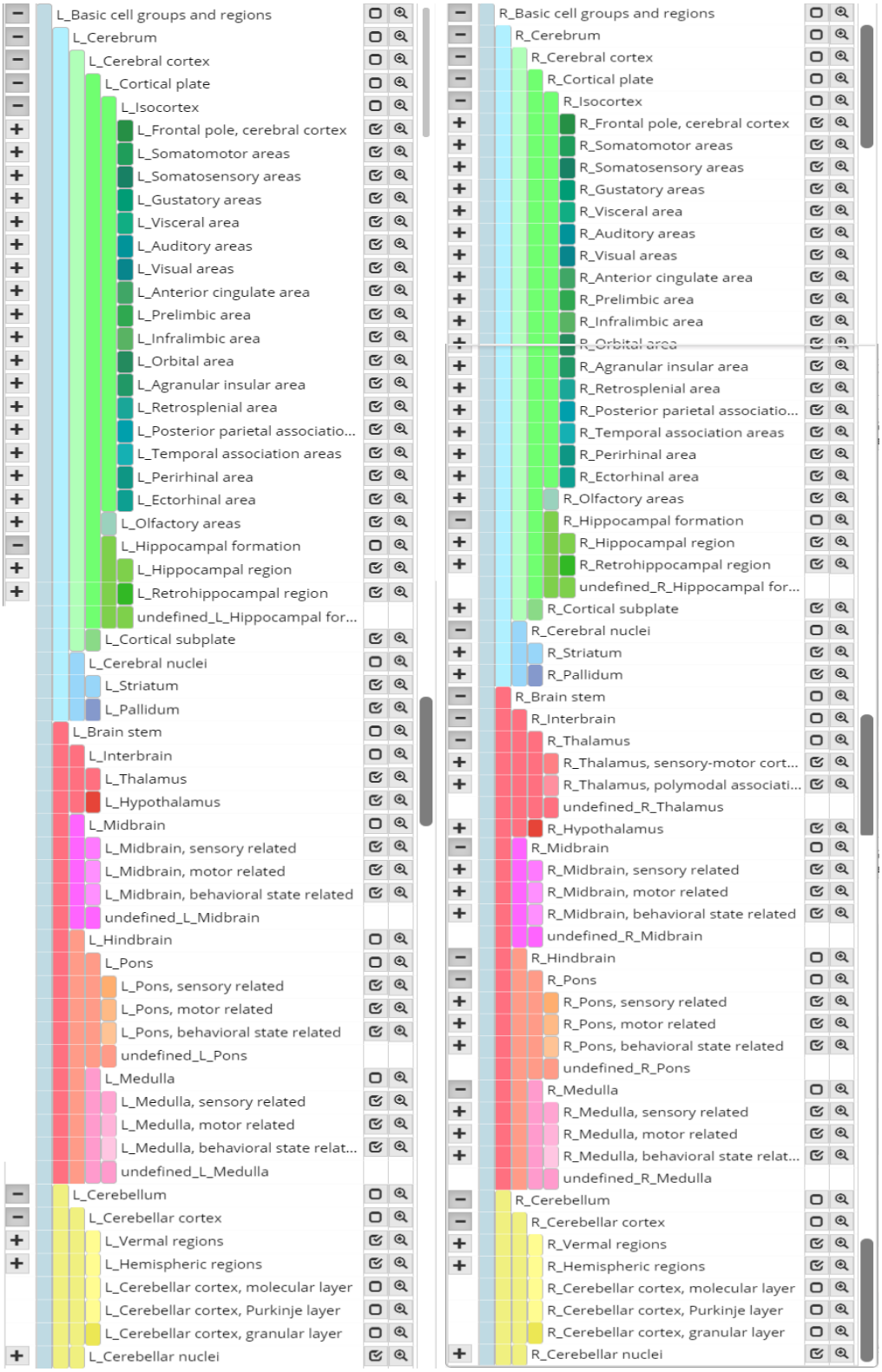
(d) "closing" right brain selection

Figure A.8: Left and right layout results



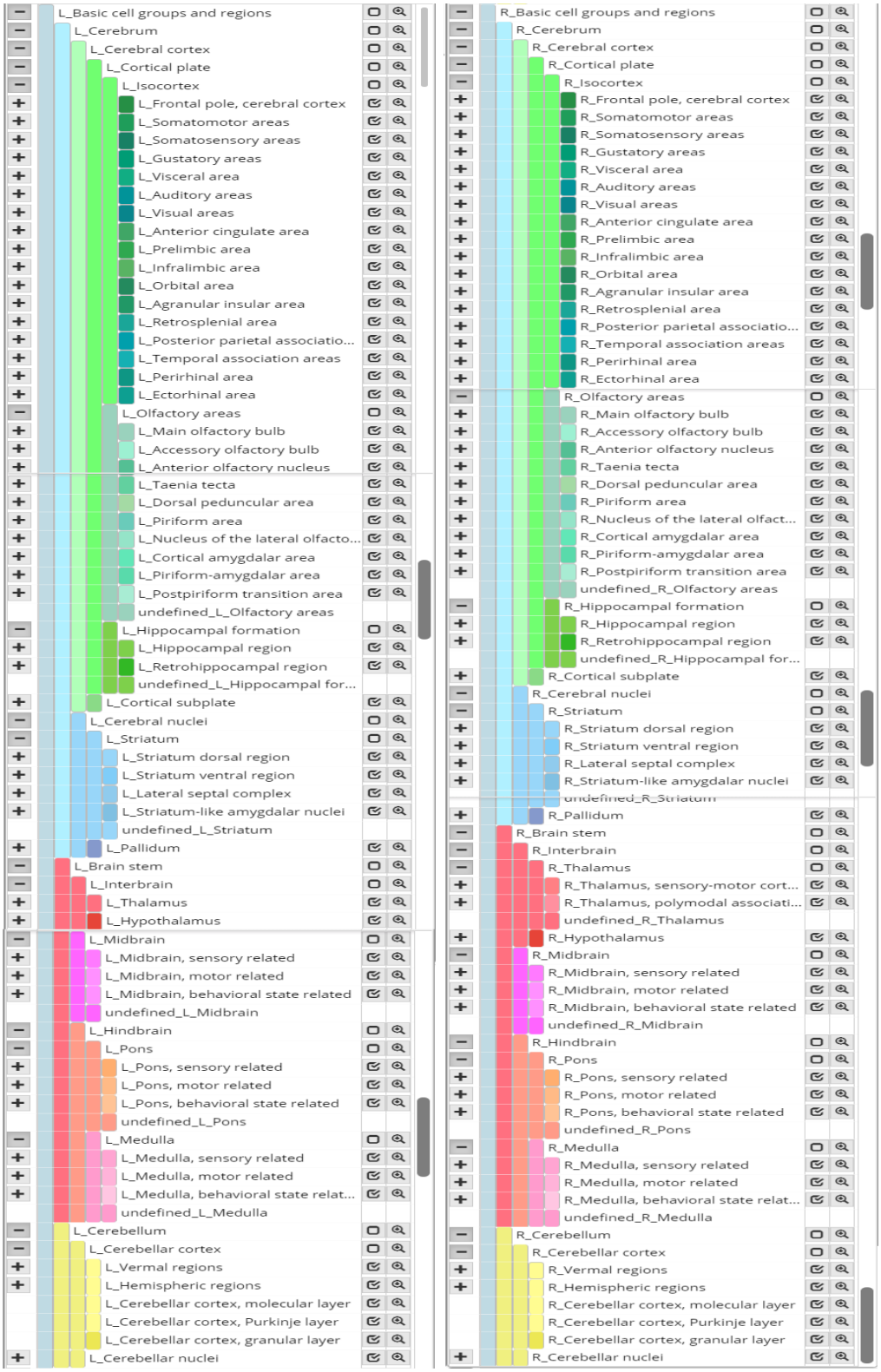
(a) 34 regions selection (b) 36 regions selection (left side of the brain)

Figure A.9: Selections shown in the parcellation browser



(a) 76 regions selection (left side) (b) 76 regions selection (right side)

Figure A.10: 76 regions selection shown in the parcellation browser



(a) 100 regions selection (left side) (b) 100 regions selection (right side)

Figure A.11: 100 regions selection shown in the parcellation browser

List of Figures

Bibliography

[all] Allen institute. https://www.alleninstitute.org/. Accessed: 2019-03-20.

[APLK+15] Russell A. Poldrack, Timothy Laumann, Oluwasanmi Koyejo, Brenda Gregory, Ashleigh Hover, Mei-Yen Chen, Krzysztof Gorgolewski, Jeffrey Luci, Sung Jun Joo, Ryan Boyd, Scott Hunicke-Smith, Zack Booth Simpson, Thomas Caven, Vanessa Sochat, James Shine, Evan Gordon, Abraham Z. Snyder, Babatunde Adeyemo, Steve Petersen, and Jeanette Mumford. Long-term neural and physiological phenotyping of a single human. *Nature Communications*, 6:8885, 12 2015.

[BD07] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Asia-Pacific Symposium on Visualisation 2007*, pages 133–140, Los Alamitos, CA, USA, feb 2007. IEEE Computer Society.

[braa] Brainmap. http://atlas.brain-map.org/. Accessed: 2019-03-20.

[brab] Human brain project. https://www.humanbrainproject.eu. Accessed: 2019-03-20.

[BSL+14] J. Böttger, A. Schäfer, G. Lohmann, A. Villringer, and D. S. Margulies. Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):471–480, March 2014.

[col] Cola github. https://github.com/cytoscape/cytoscape.js-cola. Accessed: 2019-03-20.

[cosa] Cose-bilkent github. https://github.com/cytoscape/cytoscape.js-cose-bilkent. Accessed: 2019-03-20.

[cosb] Cose github. https://github.com/cytoscape/cytoscape.js/tree/master/documentation/demos/cose-layout. Accessed: 2019-03-20.

[cyt] Cytoscape.js. http://js.cytoscape.org/. Accessed: 2019-03-20.

[DGC+09] Ugur Dogrusoz, Erhan Giral, Ahmet Cetintas, Ali Civril, and Emek Demir. A layout algorithm for undirected compound graphs. *Information Sciences*, 179(7):980 – 994, 2009.

[DKM06] Tim Dwyer, Yehuda Koren, and Kim Marriott. Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *IEEE transactions on visualization and computer graphics*, 12:821–8, 09 2006.

[EAD84] P. EADES. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[For86] S Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 313–322, New York, NY, USA, 1986. ACM.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, November 1991.

[GKHB18] Florian Ganglberger, Joanna Kaczanowska, Wulf Haubensak, and Katja Buehler. A data structure for real-time aggregation queries of big brain networks. *bioRxiv*, 2018.

[GKP+18] Florian Ganglberger, Joanna Kaczanowska, Josef M. Penninger, Andreas Hess, Katja Bühler, and Wulf Haubensak. Predicting functional neuroanatomical maps from fusing brain networks with genetic information. *NeuroImage*, 170:113 – 120, 2018. Segmenting the Brain.

[GN98] Emden R. Gansner and Stephen C. North. Improved force-directed layouts. In Sue H. Whitesides, editor, *Graph Drawing*, pages 364–373, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[GSF+18] Florian Ganglberger, Nicolas Swoboda, Lisa Frauenstein, Joanna Kaczanowska, Wulf Haubensak, and Katja Bühler. Iterative exploration of big brain network data. In Anna Puig Puig, Thomas Schultz, Anna Vilanova, Ingrid Hotz, Barbora Kozlikova, and Pere-Pau Vázquez, editors, *Eurographics Workshop on Visual Computing for Biology and Medicine (2018)*. The Eurographics Association, 2018.

[IDVH14] Andrei Irimia and John Darrell Van Horn. Systematic network lesioning reveals the core white matter scaffold of the human brain. *Frontiers in human neuroscience*, 8:51, 02 2014.

[kla] Klay github. `https://github.com/cytoscape/cytoscape.js-klay`. Accessed: 2019-03-20.

[LDTS14] Roan A. LaPlante, Linda Douw, Wei Tang, and Steven M. Stufflebeam. The connectome visualization utility: Software for visualization of human brain networks. *PLOS ONE*, 9(12):1–18, 12 2014.

[LFG+15] Huang Li, Shiaofen Fang, Joaquin Goni, Joey A. Contreras, Yanhua Liang, Chengtao Cai, John D. West, Shannon L. Risacher, Yang Wang, Olaf Sporns, Andrew J. Saykin, Li Shen, and [Authorinst]for the ADNI. Integrated visualization of human brain connectome data. In Yike Guo, Karl Friston, Faisal Aldo, Sean Hill, and Hanchuan Peng, editors, *Brain Informatics and Health*, pages 295–305, Cham, 2015. Springer International Publishing.

[MBB+17] S. Murugesan, K. Bouchard, J. A. Brown, B. Hamann, W. W. Seeley, A. Trujillo, and G. H. Weber. Brain modulyzer: Interactive visual analysis of functional brain connectivity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(4):805–818, July 2017.

[MM14] Angela M. Muller and Martin Meyer. Language in the brain at rest: new insights from resting state data and graph theoretical analysis. *Frontiers in Human Neuroscience*, 8:228, 2014.

[nod] Cose demo. `http://js.cytoscape.org/demos/cose-layout/`. Accessed: 2019-03-20.

[rea] React. `https://reactjs.org`. Accessed: 2019-03-20.

[red] Redux. `https://redux.js.org`. Accessed: 2019-03-20.

[Sor13] Johannes Sorger. neuromap - interactive graph-visualization of the fruit fly's neural circuit. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, January 2013.

[spr] Spread github. `https://github.com/cytoscape/cytoscape.js-spread`. Accessed: 2019-03-20.

[SSvH14] Christoph Daniel Schulze, Miro Spönemann, and Reinhard von Hanxleden. Drawing layered graphs with port constraints. *Journal of Visual Languages and Computing, Special Issue on Diagram Aesthetics and Layout*, 25(2):89–106, 2014.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, Feb 1981.

[XWH13] Mingrui Xia, Jinhui Wang, and Yong He. Brainnet viewer: A network visualization tool for human brain connectomics. *PLOS ONE*, 8(7):1–15, 07 2013.