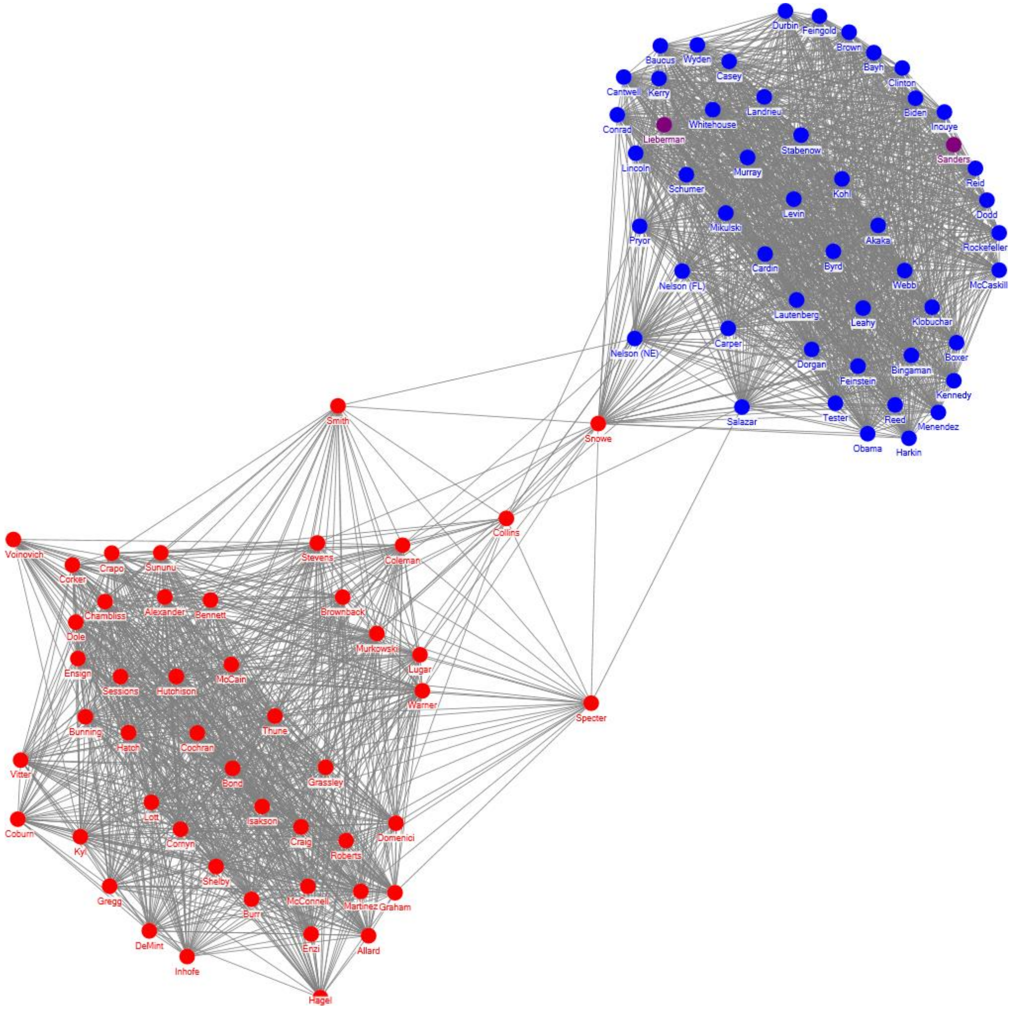
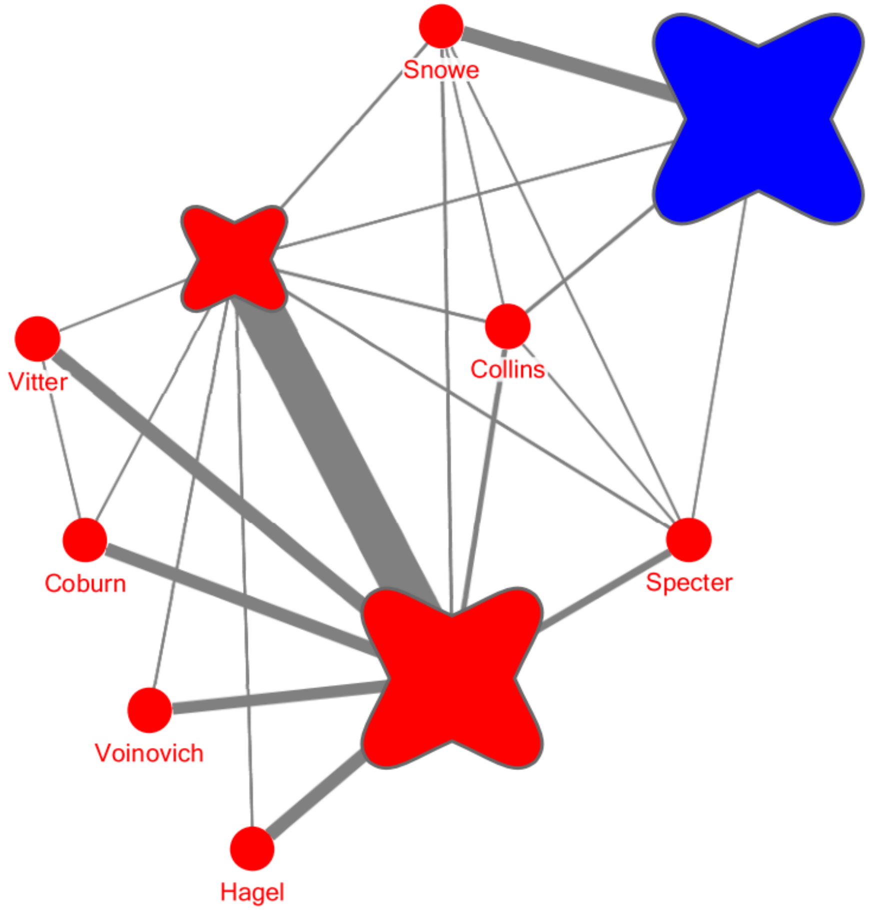


CS 7280-03 Special Topics on Visualization in Network Science

Lecture 8



Professor Cody Dunne

https://codydunne.github.io/cs7280-f16/
c.dunne@northeastern.edu

Office Hours 10/6: 10:45—11:45am

Homework 3

https://codydunne.github.io/cs7280-f16/hw/Homework-3-D3-spring-layout

Project Part 1: Initial Ideas

https://codydunne.github.io/cs7280-f16/project

Zurich Insurance – Todd Shock

- How to identify people from metadata. Zurich doesn’t always get the details from their brokers.
- Supply chain disruptions
  - How to identify concentrations of risk
  - What-if analyses
- How do you decide whether to insure a company? Look at the people…
- How do you make travel policy and get leverage with suppliers?
  - **Access to all 2015 and 2016 flight, hotel, auto, and train reservations!**

How to read and critique a paper, and keep track of it!

Reference Management

- JabRef
  - FOSS cross-platform
  - Extended by Docear
  - BibTeX native
- Zotero
  - FOSS cross-platform browser plugin
  - BibTeX export (unclean)
- Mendeley
  - Freeware, cross platform
  - Owned by Thomson Reuters
  - BibTeX export
- Papers
  - Commercial, Mac only
- Endnote
  - Advise against
  - Commercial, owned by Thomson Reuters

Discussion: TopoLayout

TopoLayout: Multi-Level Graph Layout by Topological Features

Daniel Archambault, Tamara Munzner *IEEE Member*, David Auber

***Abstract*— We describe TopoLayout, a feature-based, multi-level algorithm that draws undirected graphs based on the topological features they contain. Topological features are detected recursively inside the graph, and their subgraphs are collapsed into single nodes, forming a graph hierarchy. Each feature is drawn with an algorithm tuned for its topology. As would be expected from a feature-based approach, the runtime and visual quality of TopoLayout depends on the number and types of topological features present in the graph. We show experimental results comparing speed and visual quality for TopoLayout against four other multi-level algorithms on a variety of datasets with a range of connectivities and sizes. TopoLayout frequently improves the results in terms of speed and visual quality on these datasets.**

***Index Terms*— Information Visualization, Graphs and Networks, Graph Visualization**

I. Introduction

Recently, **multi-level** approaches for graph drawing have been studied to overcome the size and visual quality limitations of previous work. Multi-level algorithms typically construct a graph hierarchy with the original graph at the leaf level and coarser approximations at higher levels. Current multi-level approaches typically only exploit local connectivity in the graph and treat all nodes and edges similarly. The resulting drawings are uniform, but low-level structure within the high-level structure of the graph is difficult to see.

We introduce a **feature-based** approach to multi-level graph drawing. In this approach, features of interest are recursively detected in the graph and replaced with meta-nodes at a coarser level. Appropriate drawing algorithms for each feature are selected based on the type of feature detected. Our approach to feature-based, multi-level graph drawing recursively detects **topological features** such as trees, connected components, and biconnected components, which have been well studied in the literature. We also detect highly connected clusters: features of interest in power law or small world graphs. To show that we can expand our system beyond strict topological features, we detect when the high-dimensional embedder (HDE) [22] algorithm is a suitable choice for layout. HDE is an efficient algorithm for drawing a specific subset of general graphs, many of which are grids.

The primary contribution of this work is TopoLayout, the first feature-based, multi-level algorithm. Unlike previous multi-level algorithms, the graph hierarchy is drawn bottom-up, taking the space required to draw the features into account at higher levels of the graph hierarchy. Thus, all of our layout algorithms should be **area-aware**; that is, take varying node size into account. TopoLayout also introduces passes to eliminate all node-node overlaps and to reduce the number of node-edge and edge-edge crossings.

The performance of TopoLayout is compared to existing multi-level algorithms. Although TopoLayout does have its limitations, the approach is often faster and better able to illustrate low-level structure in the context of high-level graph structure.

II. Previous and Related Work

Given a general, undirected graph G consisting of N nodes and E edges, we concern ourselves with the problem of drawing G in two dimensions. Nodes are assigned two dimensional coordinates, and if two nodes share an edge it is drawn between them as a straight line.

The problem of drawing general, undirected graphs has been well studied. Before the late 1990s, the methods were primarily focused on force-directed approaches [7], [10]–[12], [20]. These methods perform well for many types of graphs, but do not scale to graphs of thousands of nodes. To overcome this limitation, multi-level approaches and approaches which rely more heavily on user interaction have been proposed. In addition, a few previous approaches do exploit topology. We also describe the HDE approach, so that our HDE detector can be understood.

In addition to the work presented here, we have also described some preliminary work on TopoLayout in a poster [2].

D. Archambault and T. Munzner are with University of British Columbia, {archam, tmm}@cs.ubc.ca
D. Auber is with University of Bordeaux I, auber@labri.fr

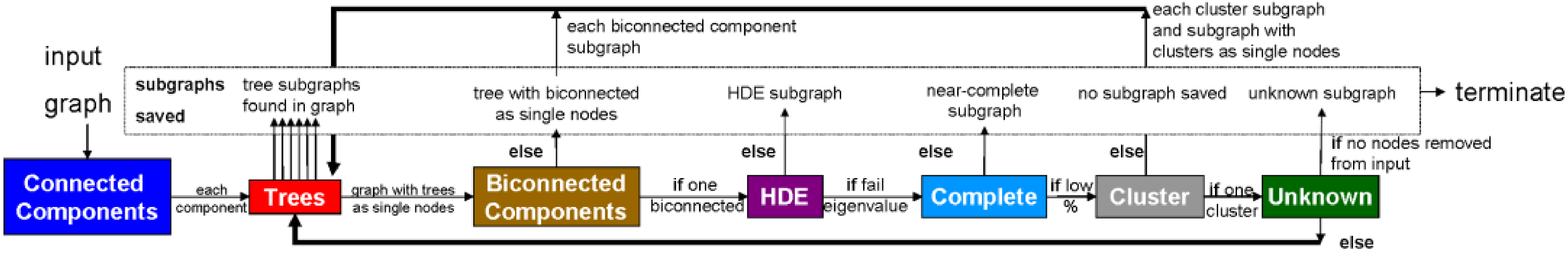
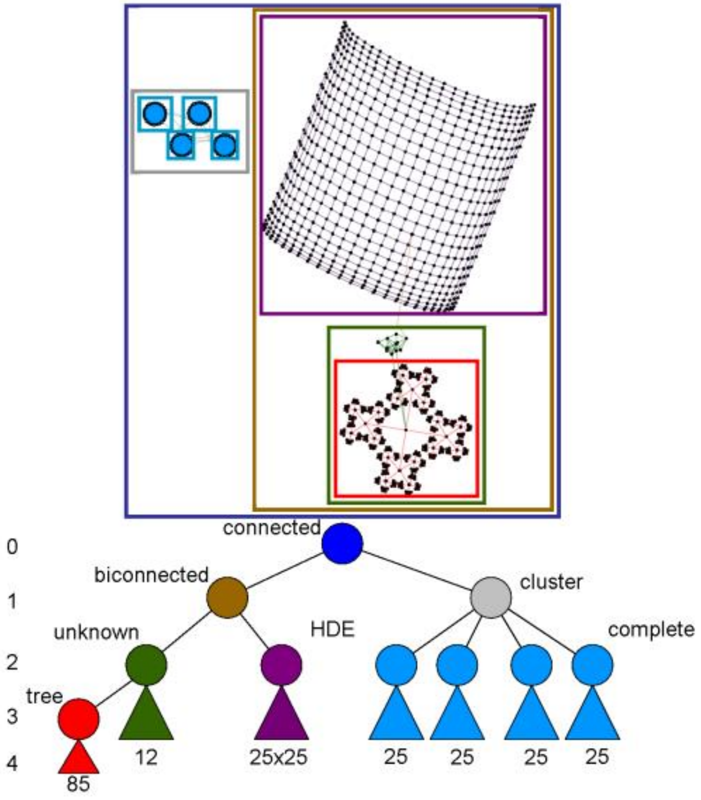
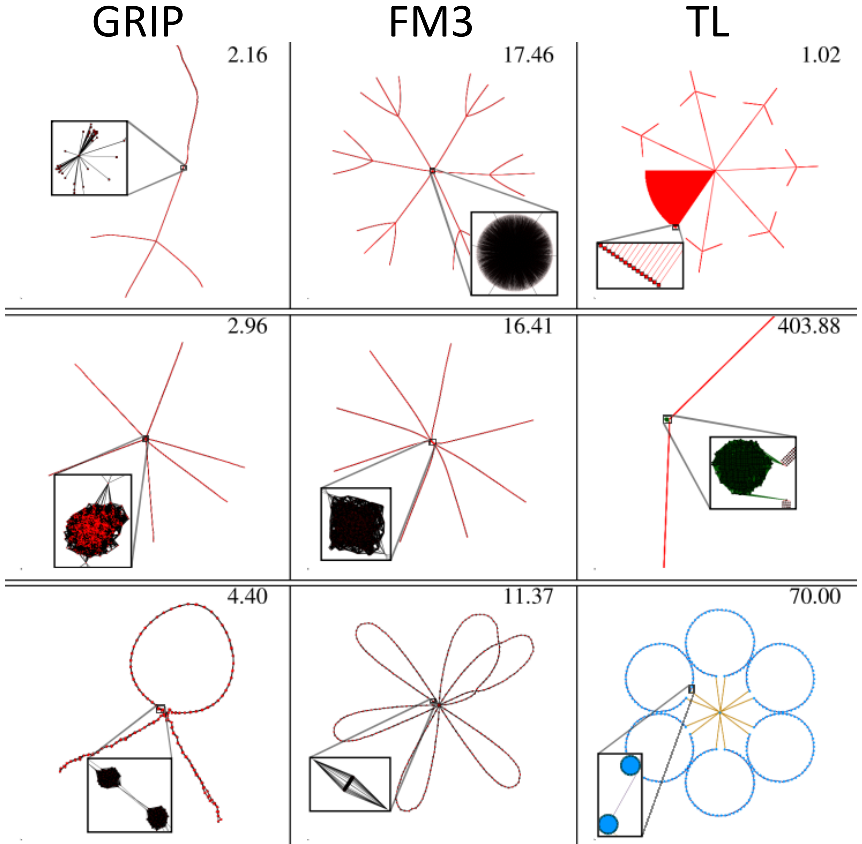


Fig. 3. Decomposition phase for TopoLayout. Detection algorithms in boxes coloured by feature type as in Figure 2. If a clause on a horizontal is true, we transition along the arrow. Otherwise, we follow the vertical arrow to save some subgraphs and recursively decompose others. Bold arrows indicate the recursive cases.





Archambault et al., 2007

Discussion: GraphMaps

GraphMaps: Browsing Large Graphs as Interactive Maps

Lev Nachmanson[1], Roman Prutkin[2], Bongshin Lee[1], Nathalie Henry Riche[1], Alexander E. Holroyd[1], and Xiaoji Chen[3]

[1] Microsoft Research, Redmond, WA, USA
{levnach,bongshin,nath,holroyd}@microsoft.com
[2] Karlsruhe Institute of Technology, Germany
roman.prutkin@kit.edu
[3] Microsoft, Redmond, WA, USA
missx@xbox.com

Abstract. Algorithms for laying out large graphs have seen significant progress in the past decade. However, browsing large graphs remains a challenge. Rendering thousands of graphical elements at once often results in a cluttered image, and navigating these elements naively can cause disorientation. To address this challenge we propose a method called GraphMaps, mimicking the browsing experience of online geographic maps.
GraphMaps creates a sequence of layers, where each layer refines the previous one. During graph browsing, GraphMaps chooses the layer corresponding to the zoom level, and renders only those entities of the layer that intersect the current viewport. The result is that, regardless of the graph size, the number of entities rendered at each view does not exceed a predefined threshold, yet all graph elements can be explored by the standard zoom and pan operations.
GraphMaps preprocesses a graph in such a way that during browsing, the geometry of the entities is stable, and the viewer is responsive. Our case studies indicate that GraphMaps is useful in gaining an overview of a large graph, and also exploring a graph on a finer level of detail.

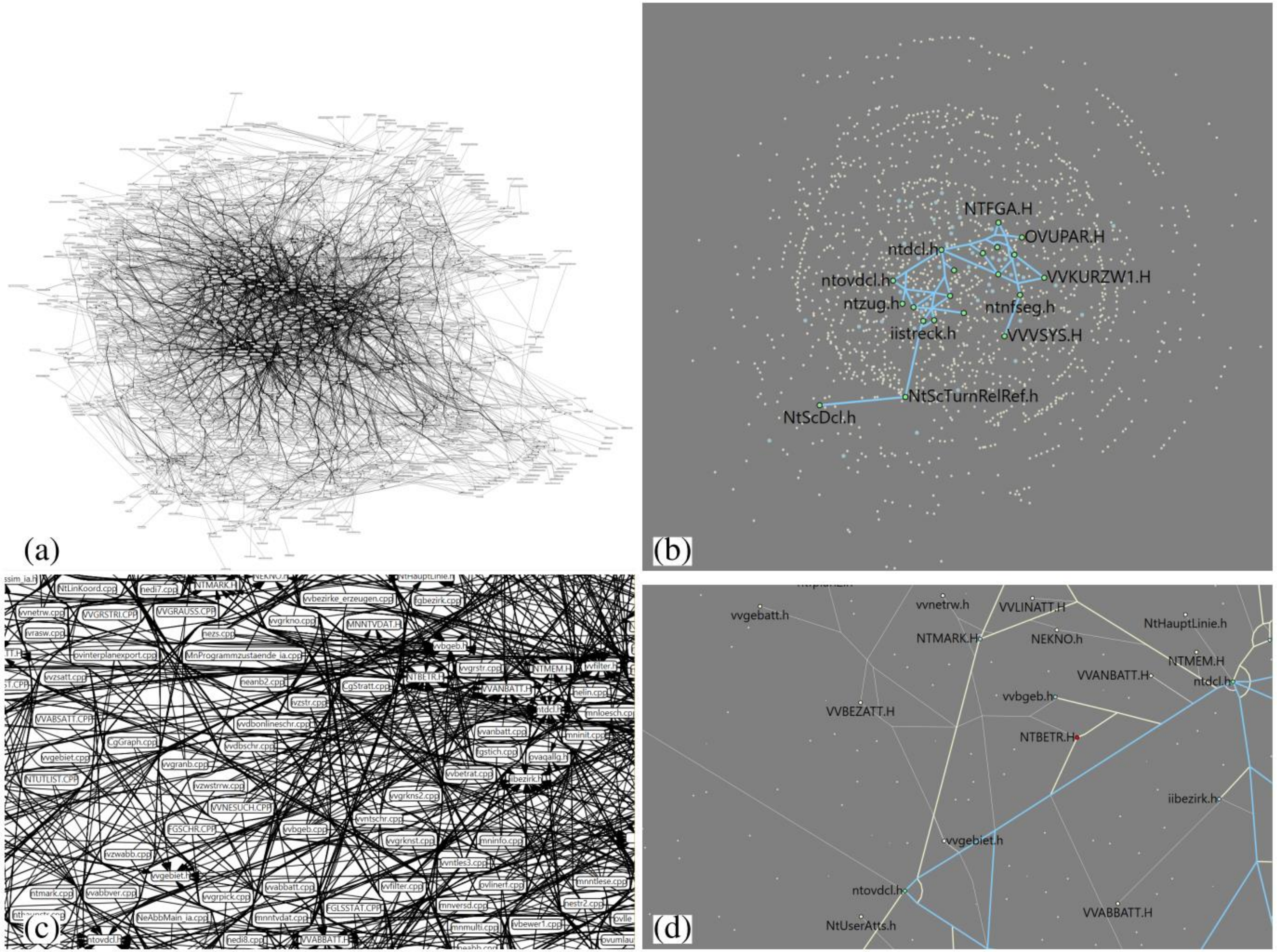
1 Introduction

Graphs are ubiquitous in many different domains such as information technology, social analysis or biology. Graphs are routinely visualized, but their large size is often a barrier. The difficulty comes not from the layout which can be calculated very fast. (For example, by using Brandes and Pich's algorithm [9] a graph with several thousand nodes and links can be laid out in a few seconds on a regular personal computer.) Rather, viewing and browsing these large graphs is problematic. Firstly, rendering thousands of graphical elements on a computer might take a considerable time and may result in a cluttered image if the graph is dense. Secondly, navigating thousands of elements rendered naively disorients the user.

Our intention is to provide a graph browsing experience similar to that of online geographic maps, for example, Bing or Google Maps. We propose a set of requirements for such a visualization and introduce a method, GraphMaps, fulfilling these requirements. GraphMaps renders a graph as an interactive map by displaying only the most

Design goals:

- Reveal most details using only zoom in, zoom out, and pan operations.
  - Assign view levels with node/edge importance
  - Interactions e.g. node/edge highlighting or search
- Mental map preservation
  - Node positions and edge trajectories should not change
- Limit visual clutter
  - Hard bound on number of rendered elements



Nachmanson et al., 2015

Discussion: DendSort

F1000Research

F1000Research 2014, 3:177 Last updated: 30 SEP 2016

SOFTWARE TOOL ARTICLE

dendsort: modular leaf ordering methods for dendrogram representations in R [version 1; referees: 2 approved]

Ryo Sakai[1,2], Raf Winand[1,2], Toni Verbeiren[1,2], Andrew Vande Moere[3], Jan Aerts[1,2]

[1]Department of Electrical Engineering (ESAT) STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, KU Leuven, 3001, Belgium
[2]iMinds Medical IT, KU Leuven, 3001, Belgium
[3]Department of Architecture, Research[x]Design, KU Leuven, 3001, Belgium

v1 **First published:** 30 Jul 2014, **3**:177 (doi: 10.12688/f1000research.4784.1)
Latest published: 30 Jul 2014, **3**:177 (doi: 10.12688/f1000research.4784.1)

Abstract

Dendrograms are graphical representations of binary tree structures resulting from agglomerative hierarchical clustering. In Life Science, a cluster heat map is a widely accepted visualization technique that utilizes the leaf order of a dendrogram to reorder the rows and columns of the data table. The derived linear order is more meaningful than a random order, because it groups similar items together. However, two consecutive items can be quite dissimilar despite proximity in the order. In addition, there are 2^{n-1} possible orderings given n input elements as the orientation of clusters at each merge can be flipped without affecting the hierarchical structure. We present two modular leaf ordering methods to encode both the monotonic order in which clusters are merged and the nested cluster relationships more faithfully in the resulting dendrogram structure. We compare dendrogram and cluster heat map visualizations created using our heuristics to the default heuristic in R and seriation-based leaf ordering methods. We find that our methods lead to a dendrogram structure with global patterns that are easier to interpret, more legible given a limited display space, and more insightful for some cases. The implementation of methods is available as an R package, named "dendsort", from the CRAN package repository. Further examples, documentations, and the source code are available at [https://bitbucket.org/biovizleuven/dendsort/].

Open Peer Review

Referee Status: ✔ ✔

	Invited Referees 1	Invited Referees 2
version 1 published 30 Jul 2014	✔ report	✔ report

1 **Eamonn Maguire**, University of Oxford UK, **Rodrigo Santamaria**, University of Salamanca Spain

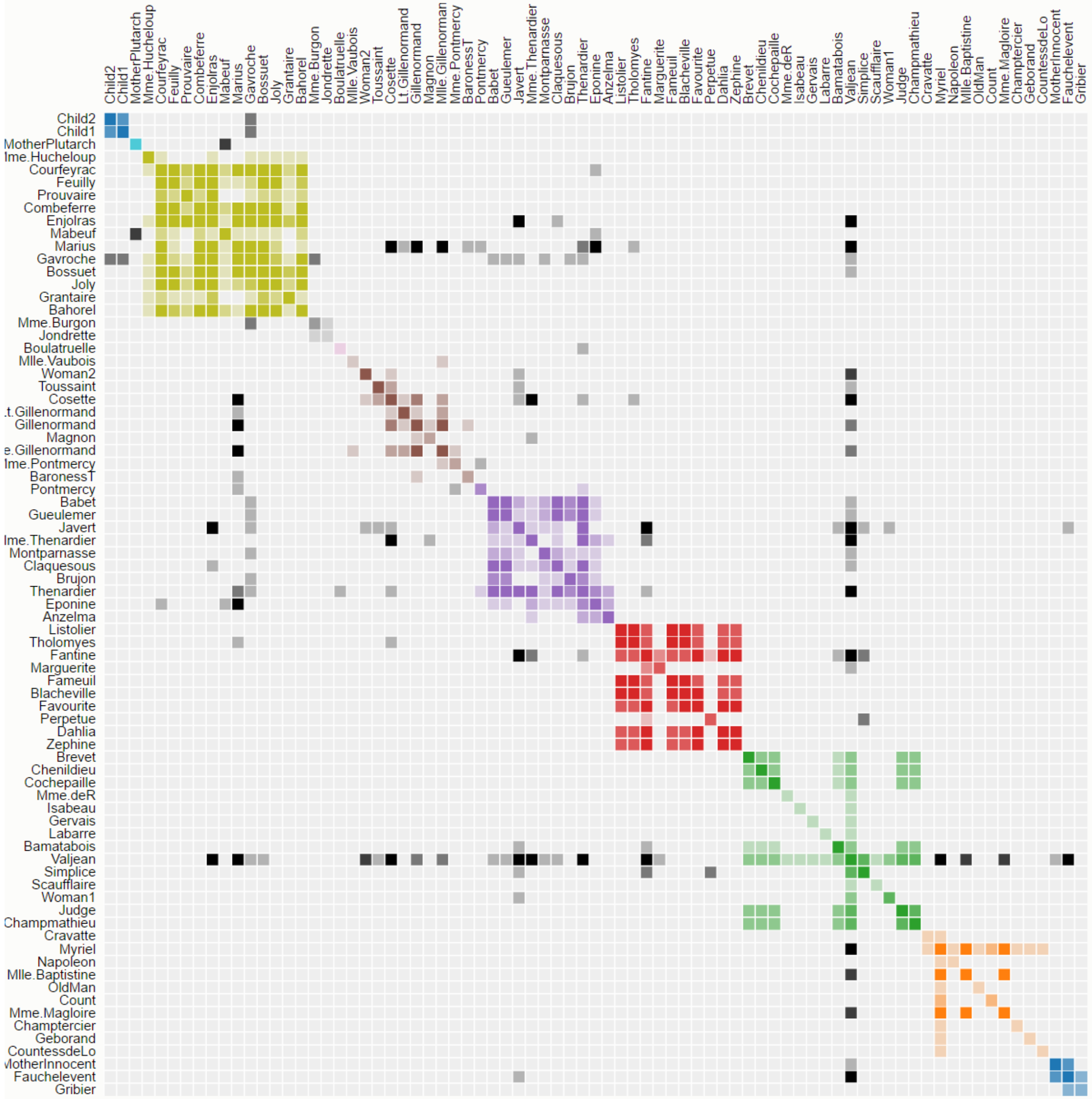
2 **Jan Oosting**, Leiden University Medical Center Netherlands

Discuss this article

Comments (0)

This article is included in the RPackage channel.

Les Misérables Co-occurrence



Order: by Cluster ▾

This matrix diagram visualizes character co-occurrences in Victor Hugo's *Les Misérables*.

Each colored cell represents two characters that appeared in the same chapter; darker cells indicate characters that co-occurred more frequently.

Use the drop-down menu to reorder the matrix and explore the data.

Built with d3.js.

Source: The Stanford GraphBase.

Mike Bostock, 2012

WDA-LS clustered co-occurrence

Use the drop-down menu to reorder the matrix and explore the data.

When ordered by cluster, rows and columns are clustered by affinity values using hierarchical agglomerative clustering.
Distance measure: Euclidean.
Linkage technique: Single.

Rows and columns are then arranged using leaf reordering using the algorithm from: Sakai, Ryo, et al. "Dendsort: modular leaf ordering methods for dendrogram representations in R." F1000Research 3 (2014).

Cell labels show count and color shows normalized affinity.

Cody Dunne and Tim Stutts, IBM Watson Health Cognitive Visualization Lab

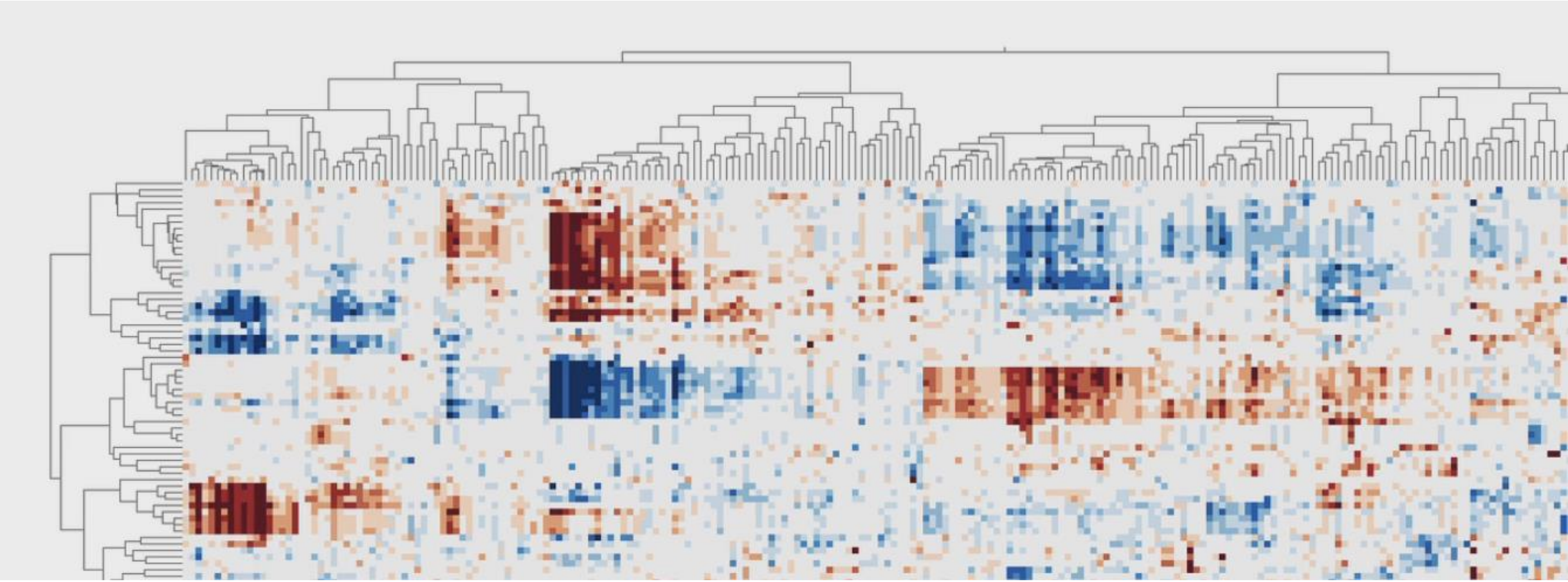
---

Dataset: genes/genes Medline (example) ▾ Edge List

Order: by Cluster ▾

The query was for genes related to the genes SOX9, TCF7L1, SMAD4, PIK3CA, KRAS in Medline.

	SOX9	TCF7L1	SMAD4	KRAS	PIK3CA
tp53	33	4	406	1295	726
apc	10	1	106	255	91
kras	10	1	166	11277	926
nras	0	0	20	878	269
hras	0	0	9	659	107
f2	2	0	5	407	0
raf1	3	1	12	760	266
alk	0	0	11	339	126
ns2	0	0	0	228	0
sos1	0	0	0	286	8
hspb3	0	0	4	279	9
ptpn11	0	0	6	192	21
cd8a	4	0	7	190	25
cd4	0	0	11	152	34
ifng	0	0	14	118	12
myc	18	1	50	278	80
mlh1	0	1	34	190	50
smad4	13	1	3052	166	53
smad2	21	1	828	12	12
smad3	20	0	658	6	12
smad7	5	0	281	0	0
smad1	17	0	262	0	6
tgfb1	23	0	230	16	7
inhbe	12	0	164	0	0
tgfbr2	5	0	123	22	6
cdkn2a	13	0	222	330	150



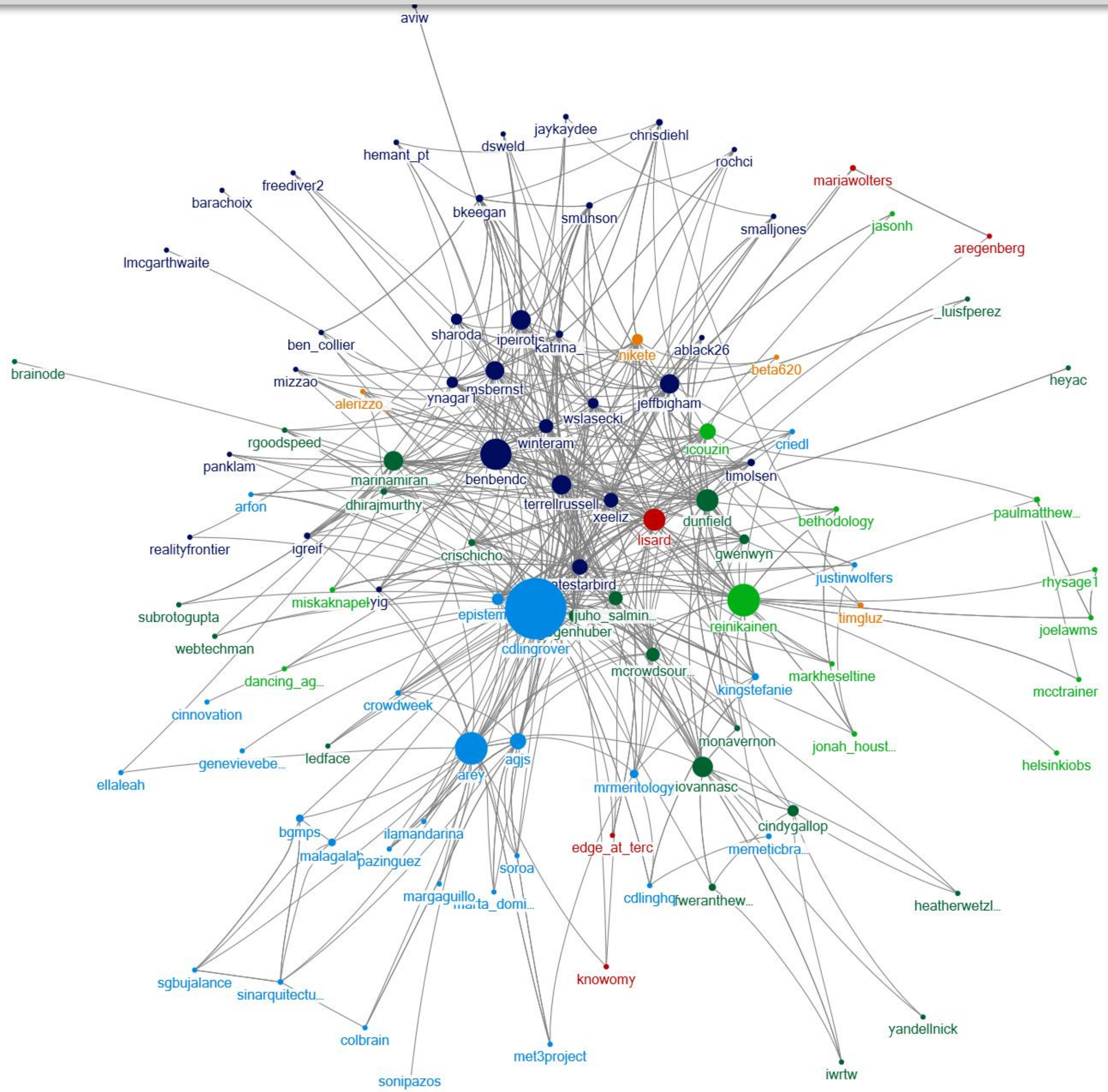
Sakai et al., 2014

Showing Group Membership

Disjoint Set Visualization

Network grouping/partitions

- Attributes
- Topology
- Combinations
- Manual

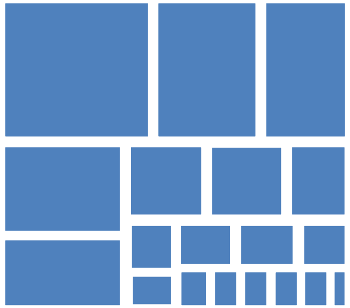


Twitter ties at the 2012
Collective Intelligence
Conference @ MIT

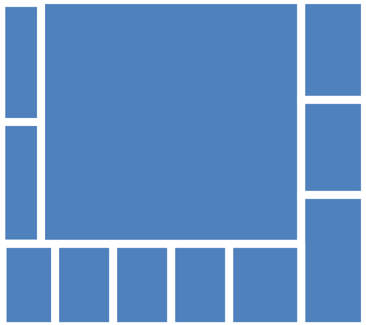
Group-in-a-Box Meta-Layouts

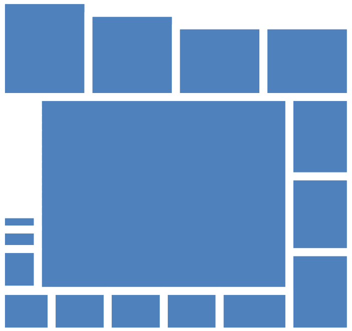
Variants

- Squarified Treemap (Rodrigues et al., 2011)

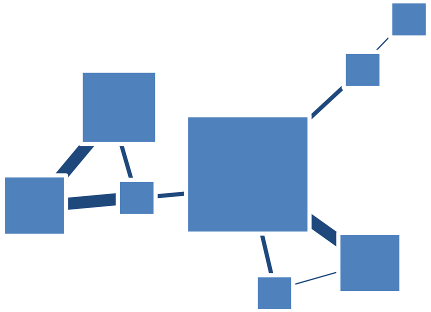


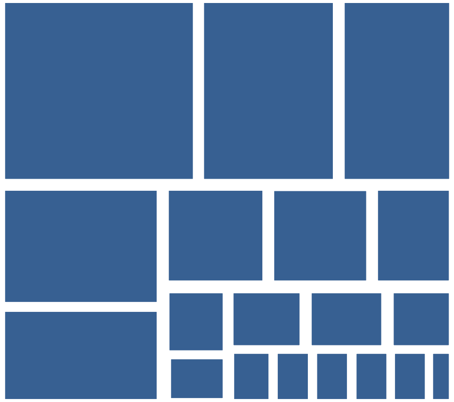
- Croissant-Doughnut



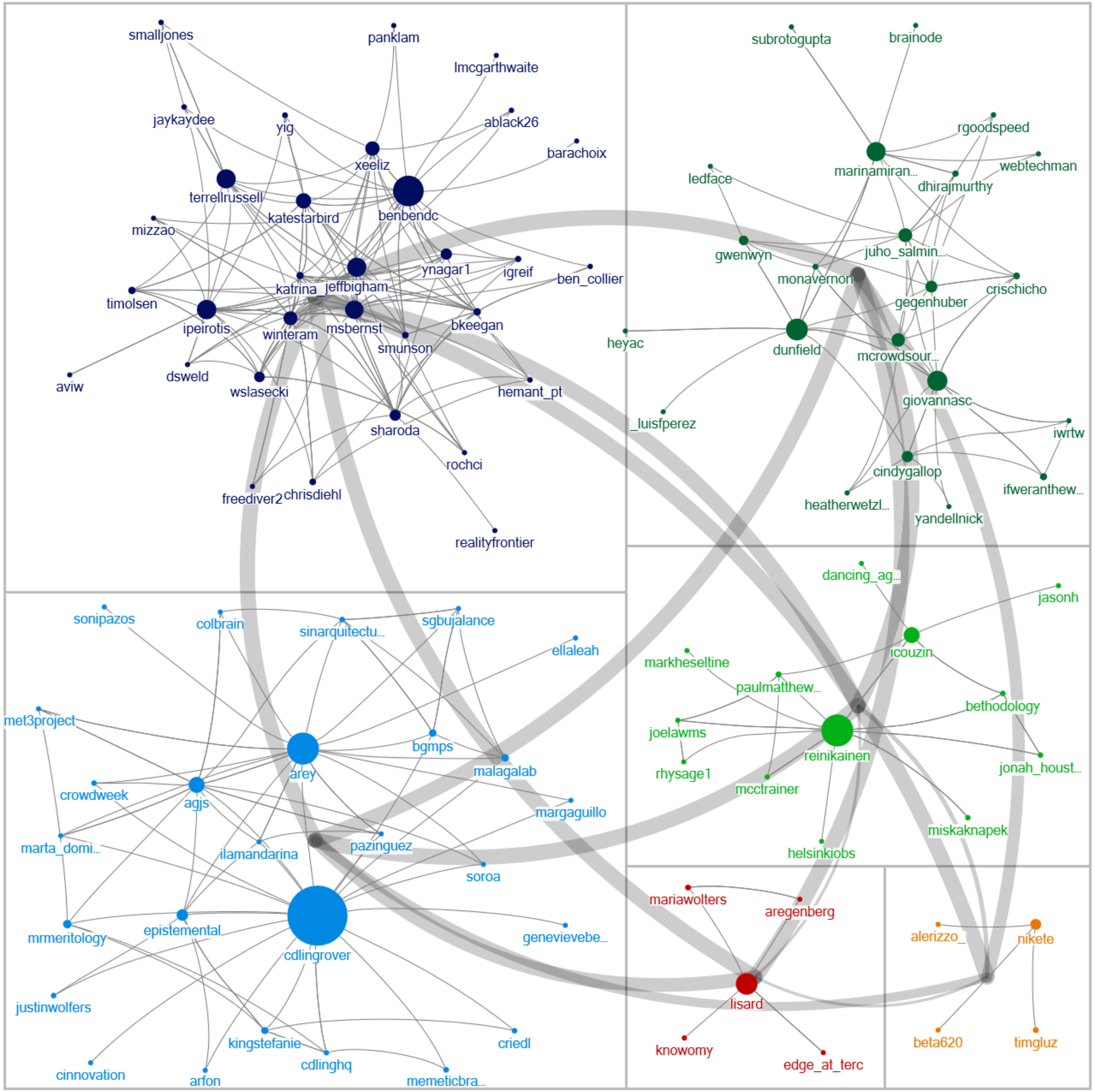


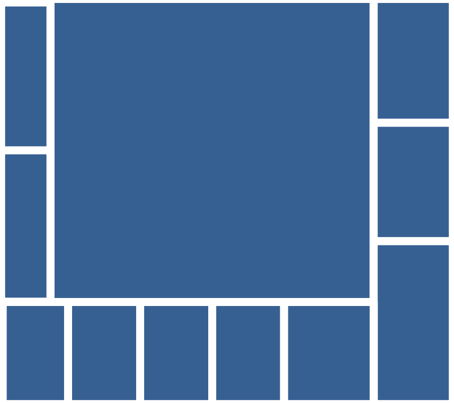
- Force-Directed



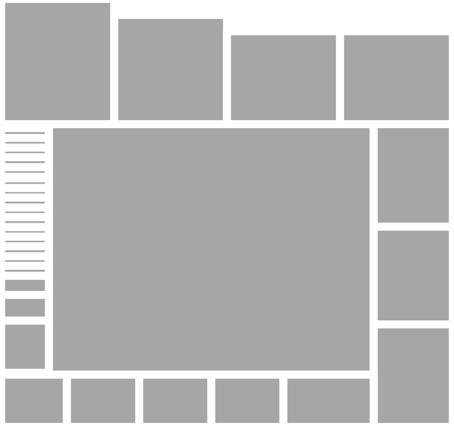


Squarified Treemap
(Rodrigues et al., 2011)

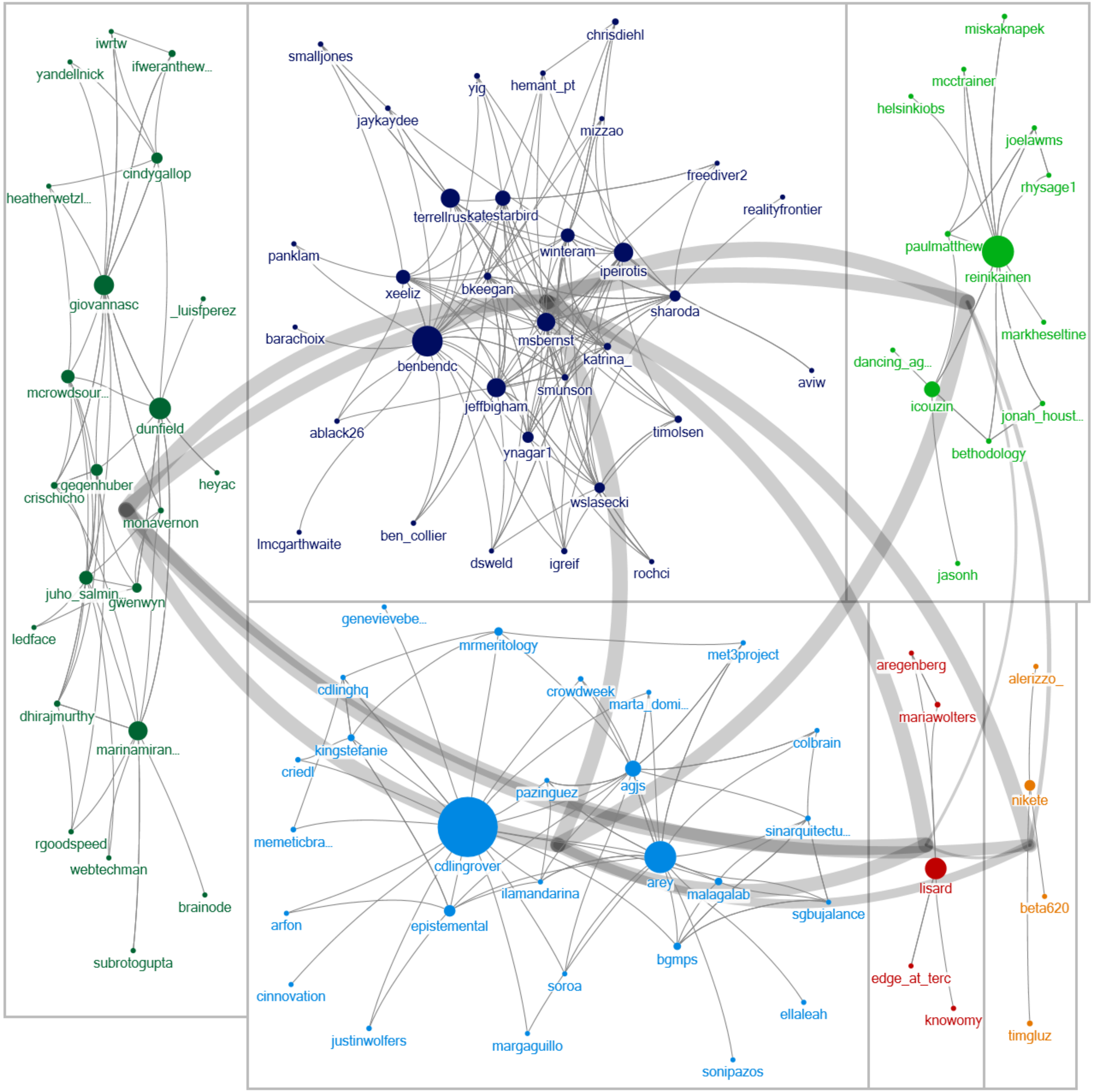


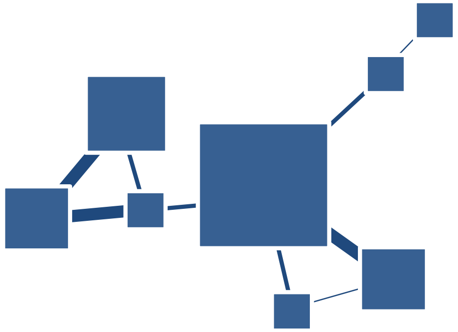


Croissant

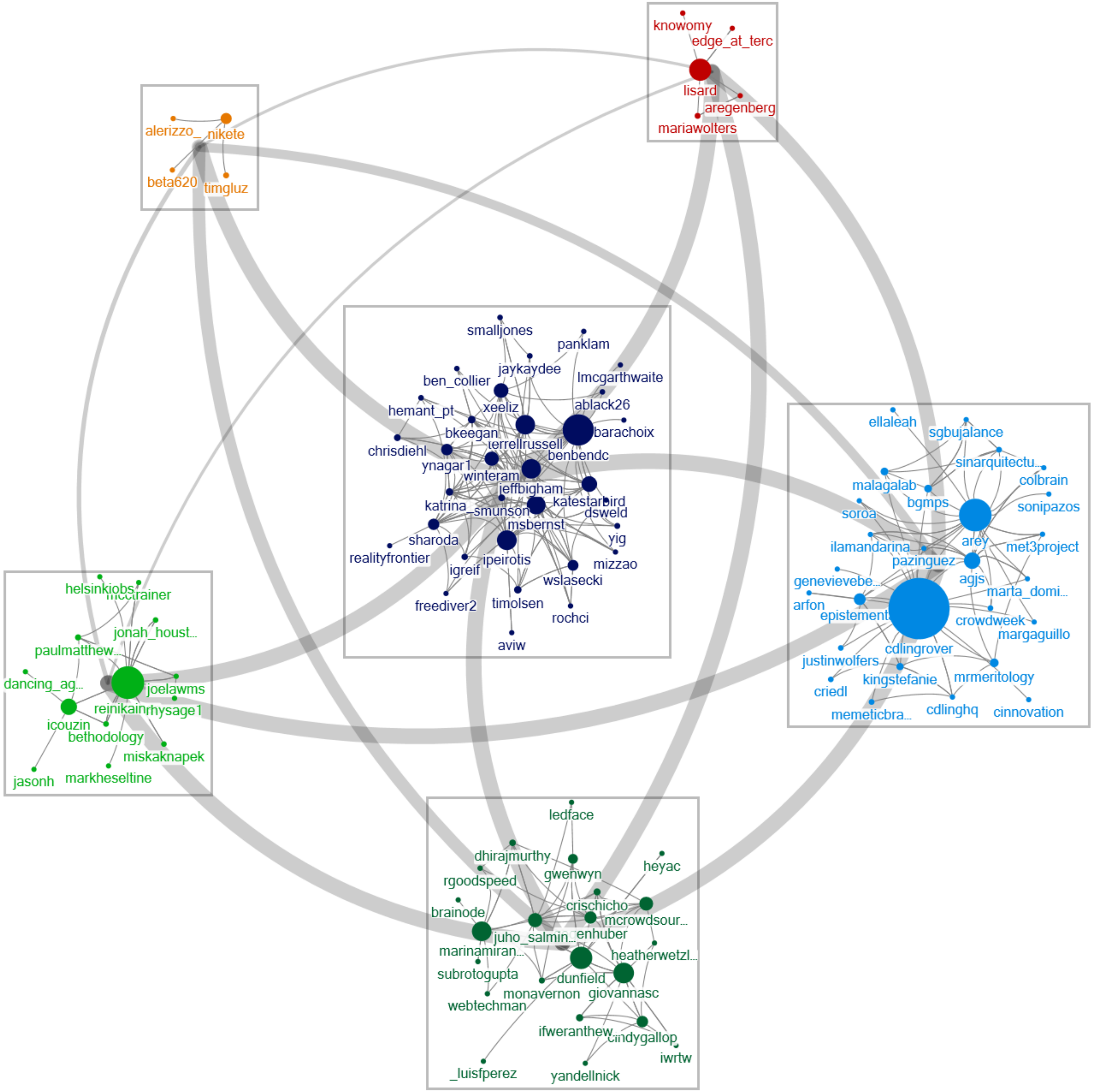


Doughnut



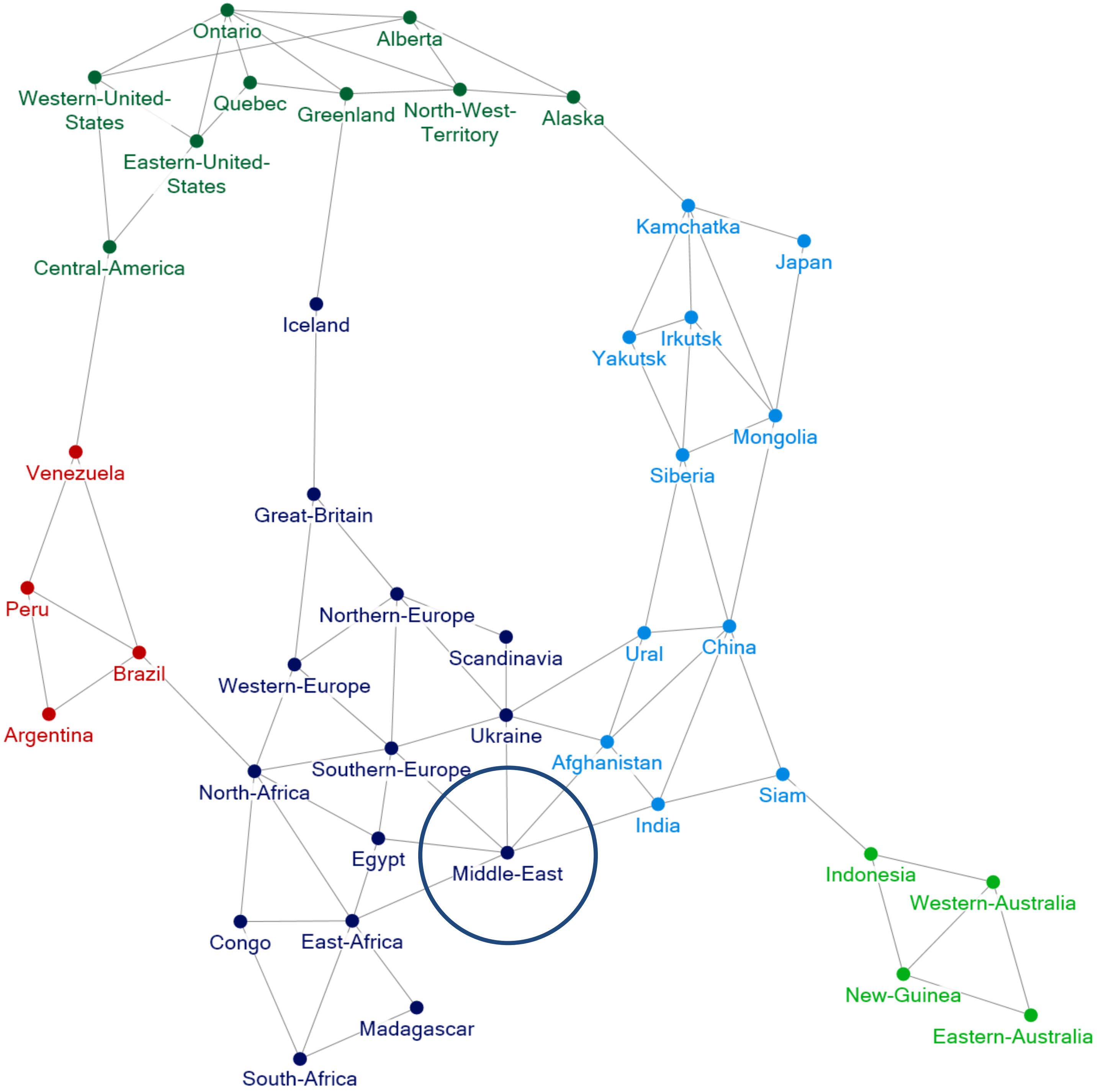


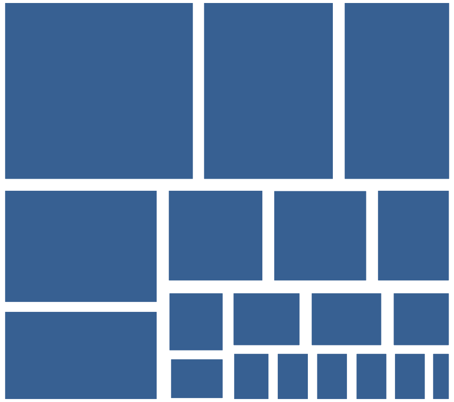
Force-Directed



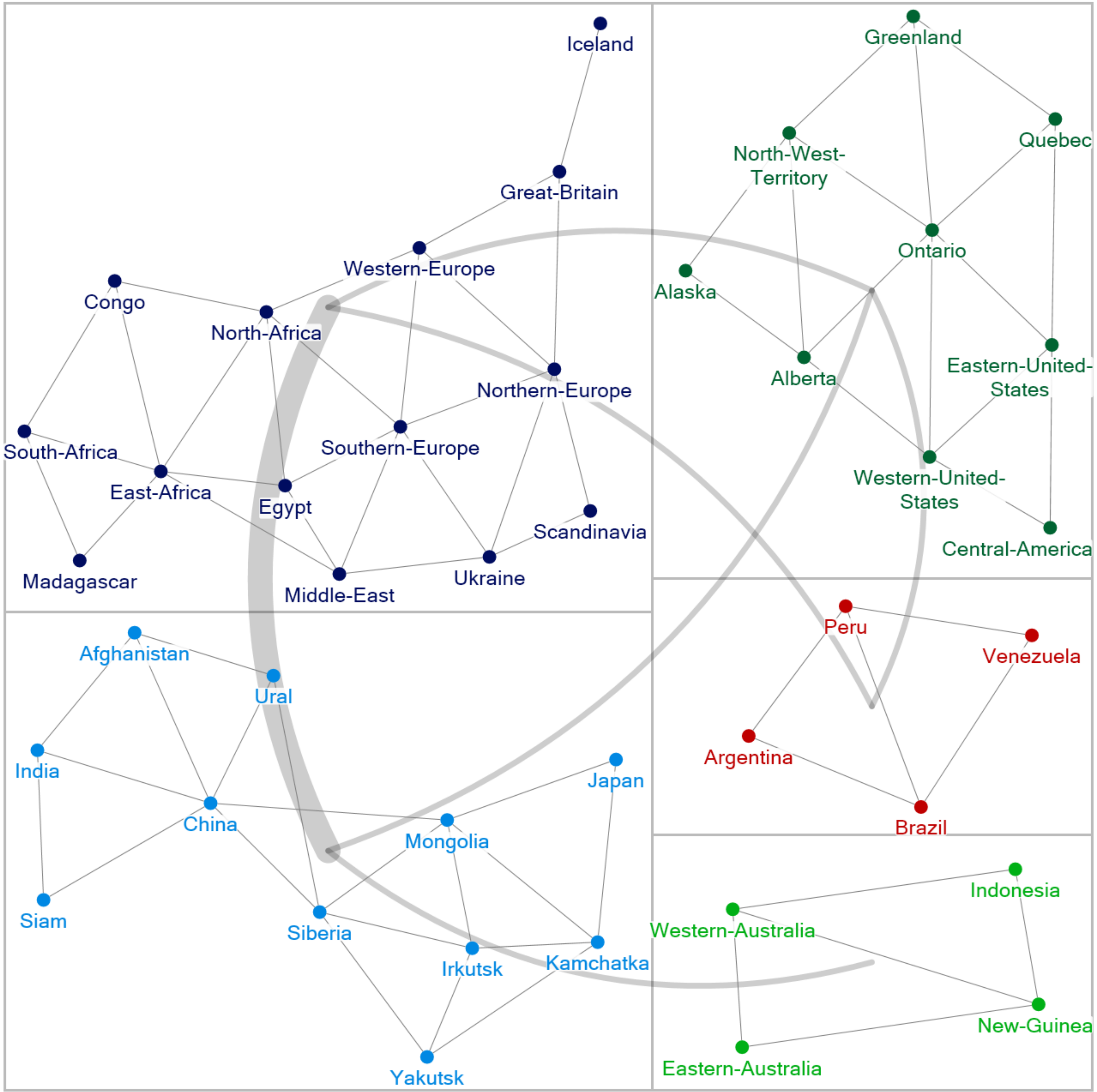

RISK
The Game of Global Domination
RISK
The Game of Global Domination
RISK
The Game of Global Domination

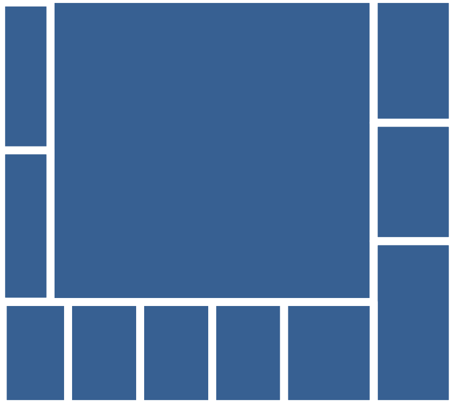
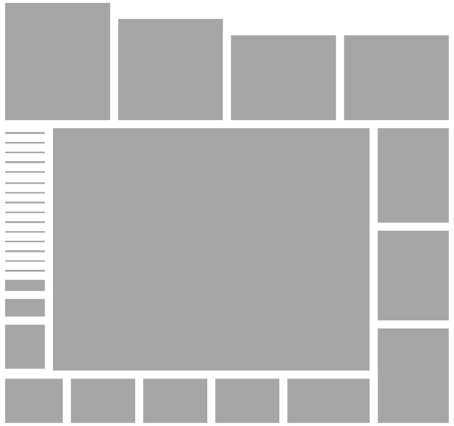
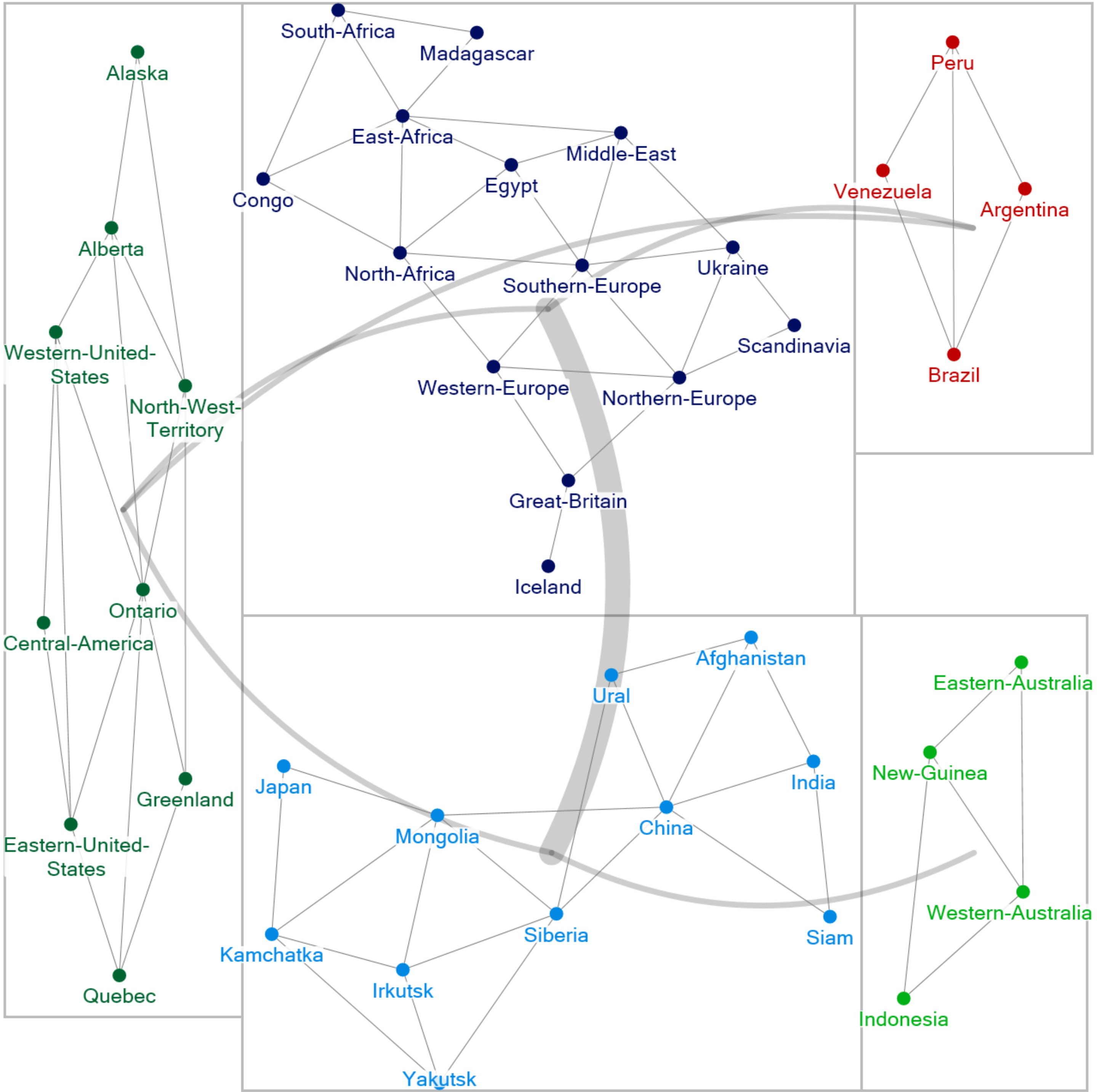
Risk Movements
Plain Layout
with Clusters

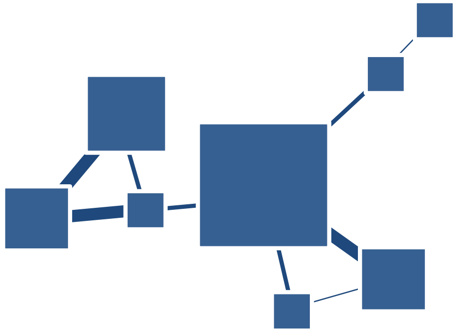




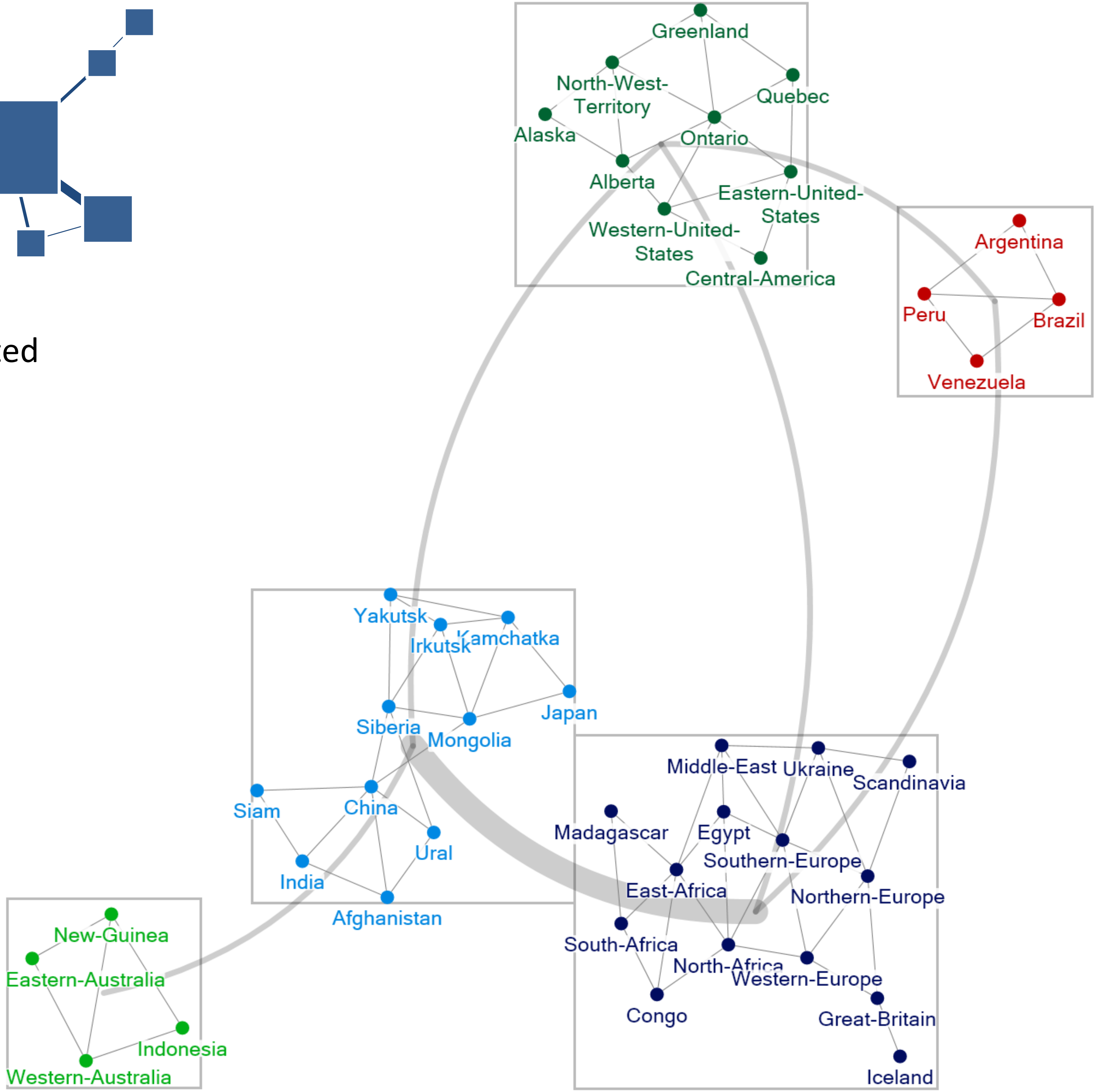
Squarified Treemap
(Rodrigues et al., 2011)

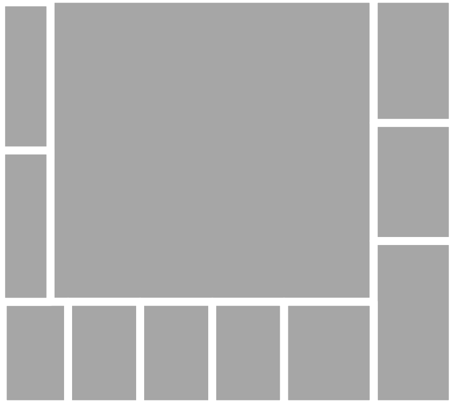



Croissant

Doughnut

Alaska
Alberta
Western-United-States
North-West-Territory
Ontario
Central-America
Greenland
Eastern-United-States
Quebec
South-Africa
Madagascar
East-Africa
Middle-East
Egypt
Congo
North-Africa
Southern-Europe
Ukraine
Scandinavia
Western-Europe
Northern-Europe
Great-Britain
Iceland
Peru
Venezuela
Argentina
Brazil
Afghanistan
Ural
India
Japan
Mongolia
China
Siberia
Siam
Kamchatka
Irkutsk
Yakutsk
Eastern-Australia
New-Guinea
Western-Australia
Indonesia

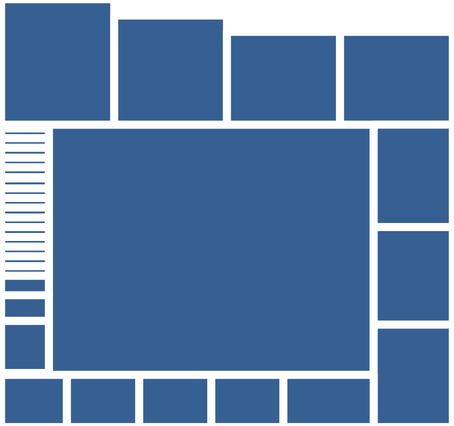


Force-Directed

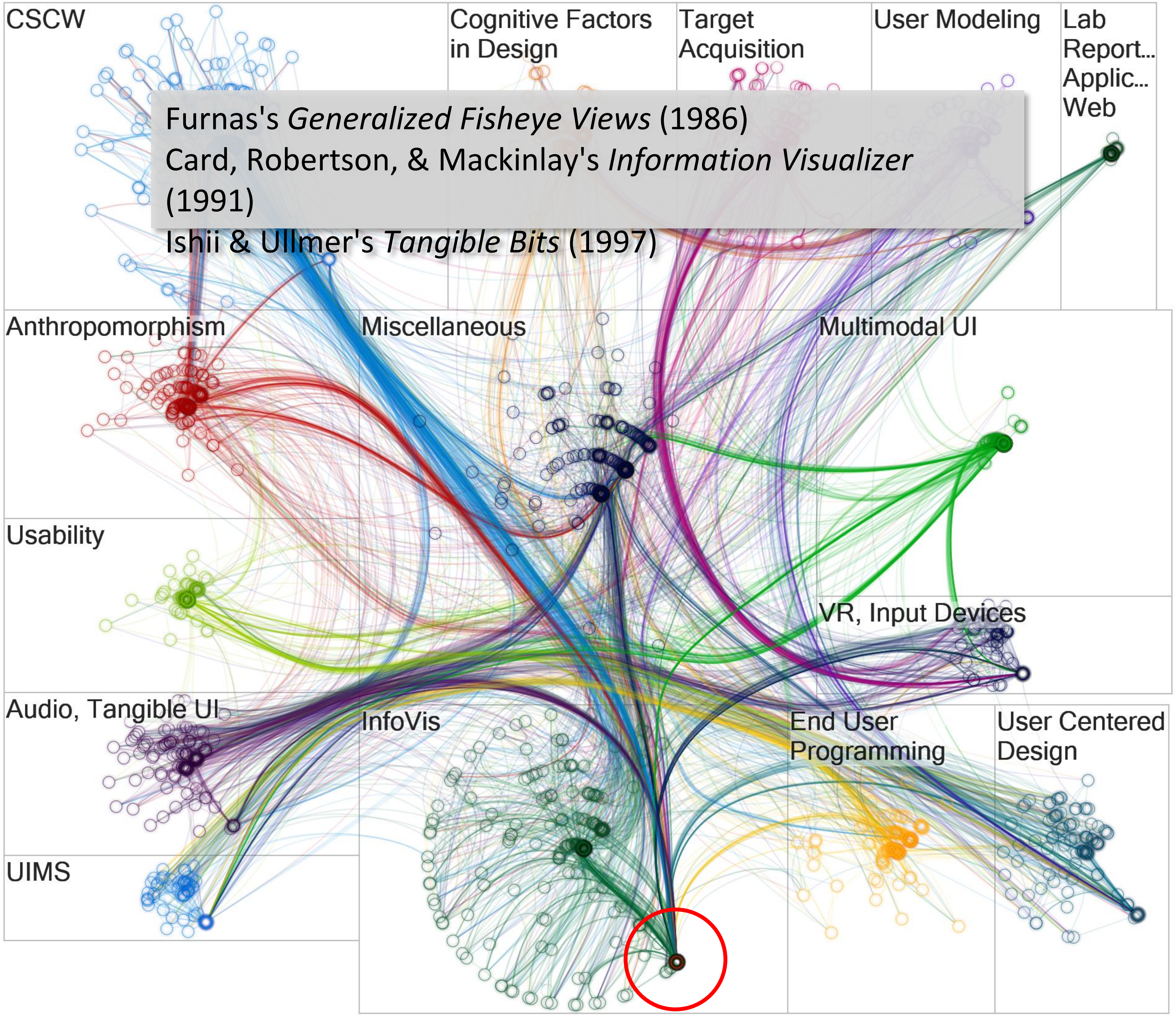


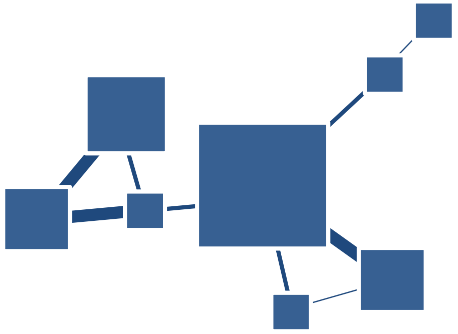


Croissant

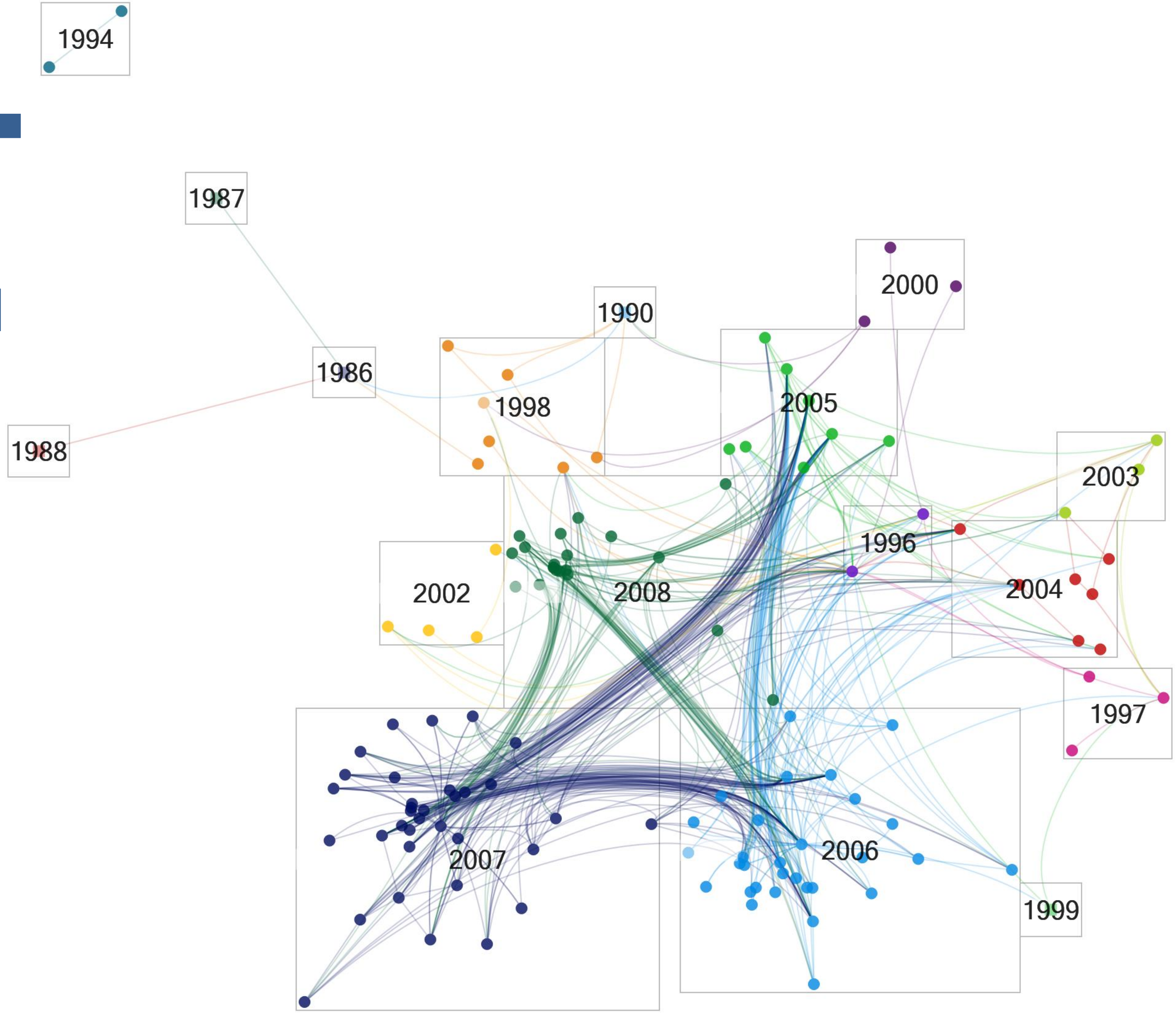


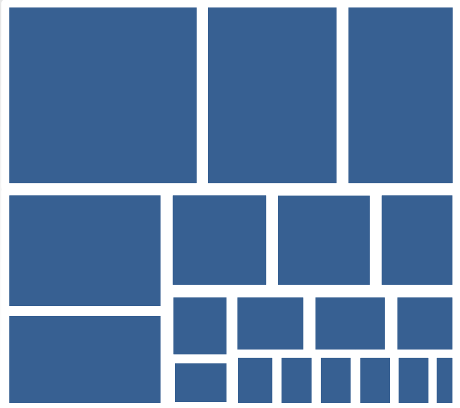
Doughnut



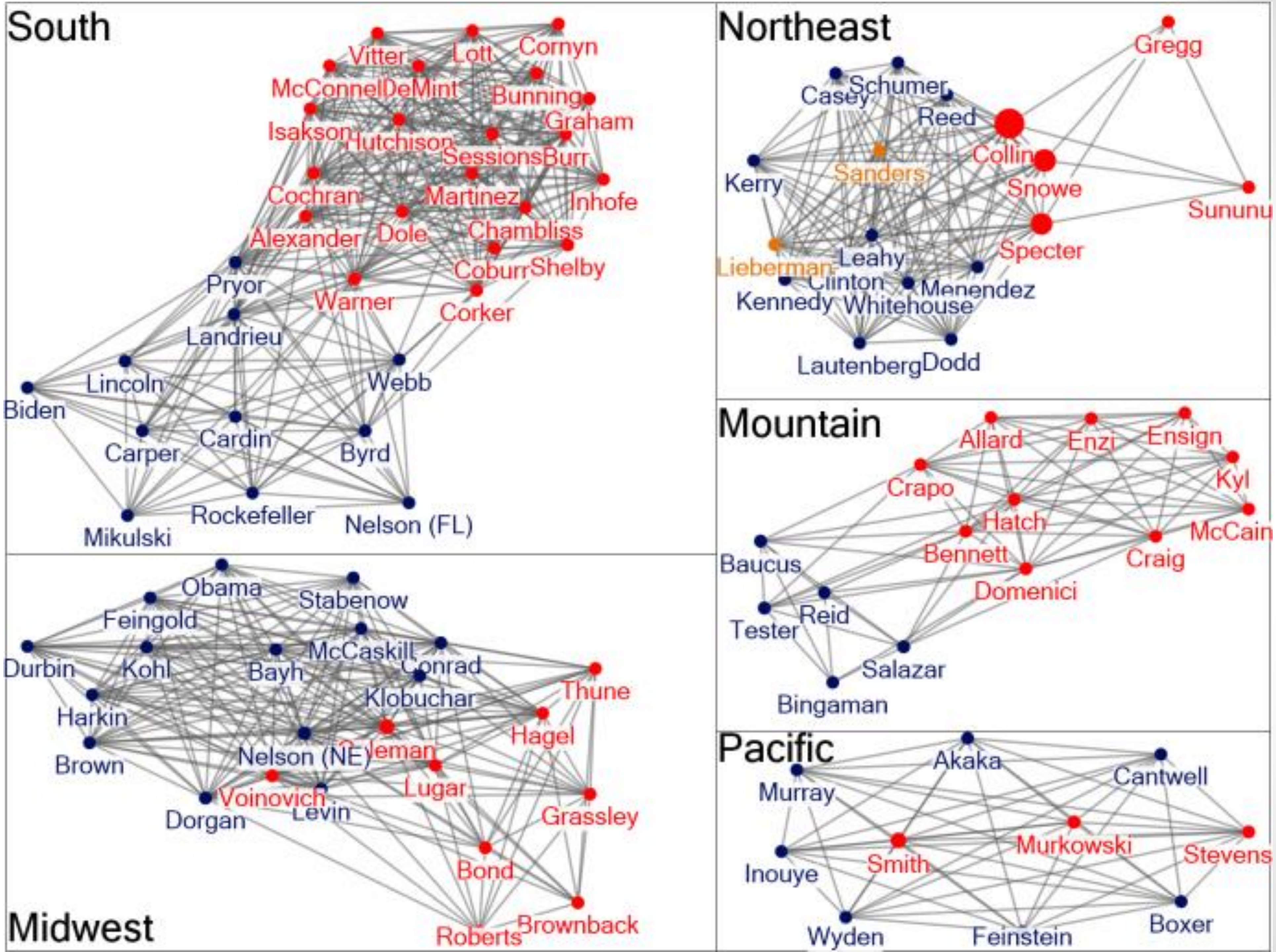


Force-Directed



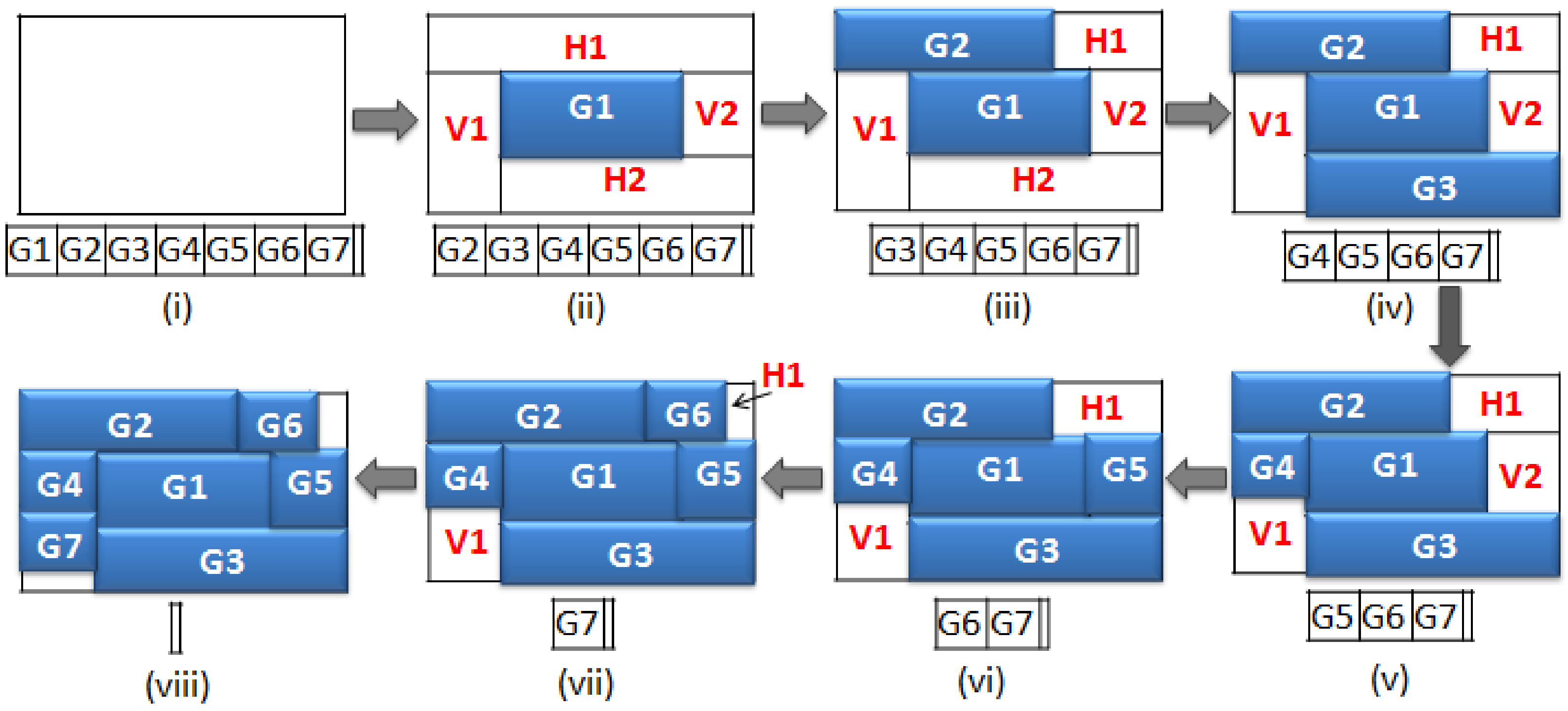


Squarified Treemap
(Rodrigues et al., 2011)



GIB Croissant-Doughnut: The Doughnut

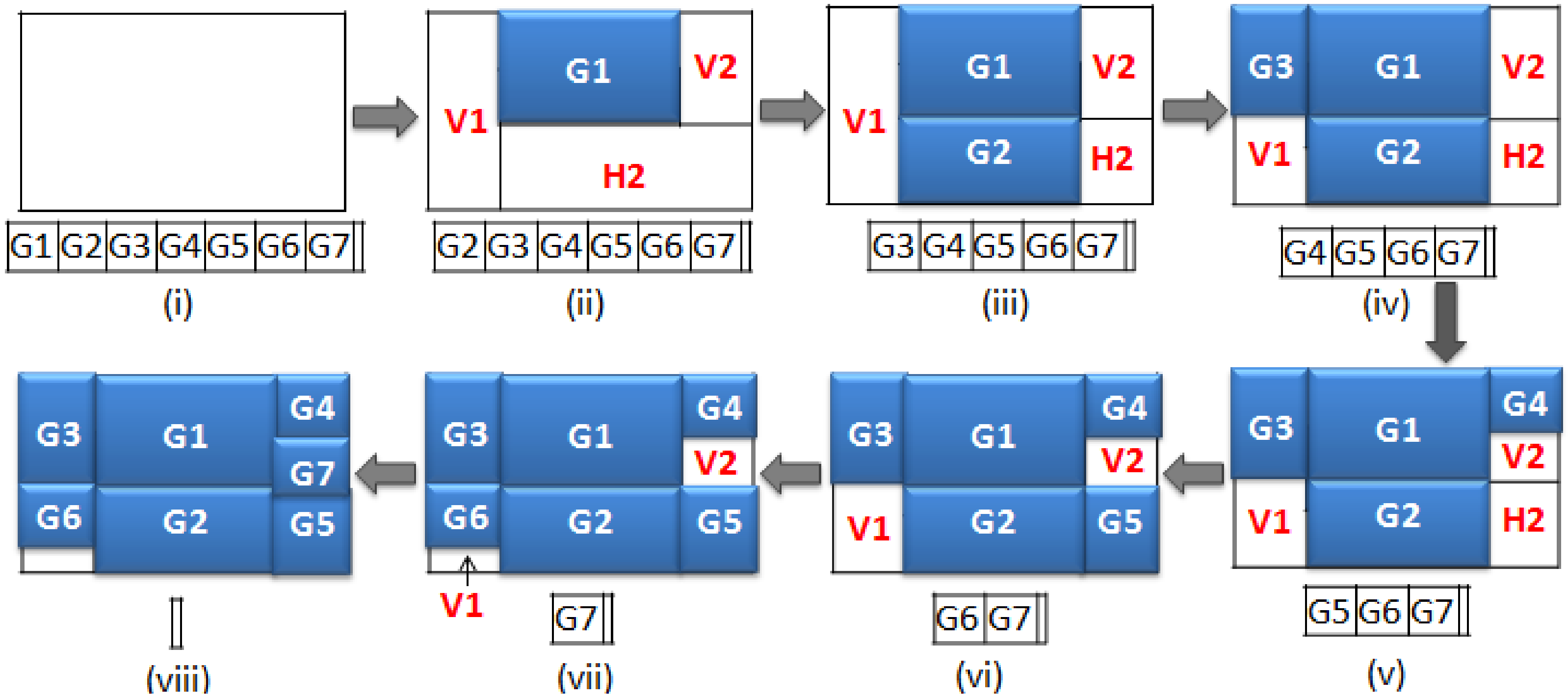
Group	Area
G1	15
G2	12
G3	14
G4	4
G5	6
G6	4
G7	4



Group Area = a * width * height * n / N

GIB Croissant-Doughnut:
The Croissant

Group	Area
G1	20
G2	15
G3	8
G4	4
G5	6
G6	4
G7	4



Group Area = a * width * height * n / N

GIB Croissant-Doughnut: Choosing Between

Definitions:

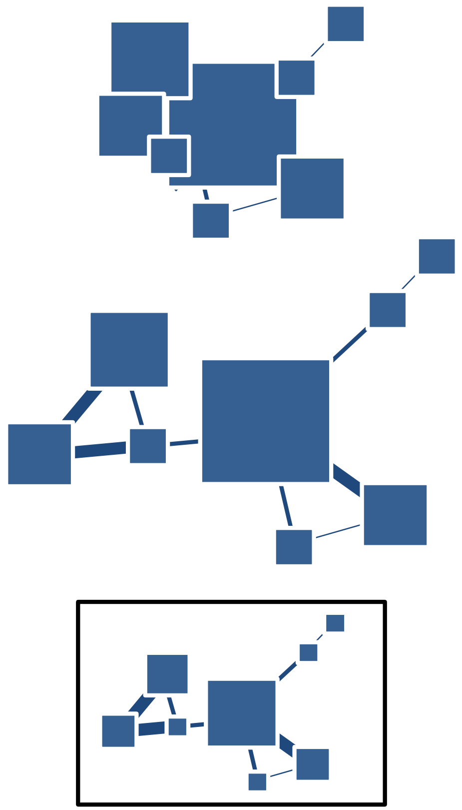
- **G**: Number of groups in network
- **G-degree**: For a group, the number of connected groups
- **G-skewness**: Fraction of nodes that are members of two most connected groups (highest G-degree).

Empirically determined values:

- **Case1: G <= 3 or G-skewness < 0.1**: Layout the group boxes using the ST-GIB layout
- **Case2: G > 3 and 0.1 <= G-skewness <= 0.45**: Layout the group boxes using Doughnut layout
- **Case3: G > 3 and G-skewness > 0.45**: Layout the group boxes using Croissant layout.

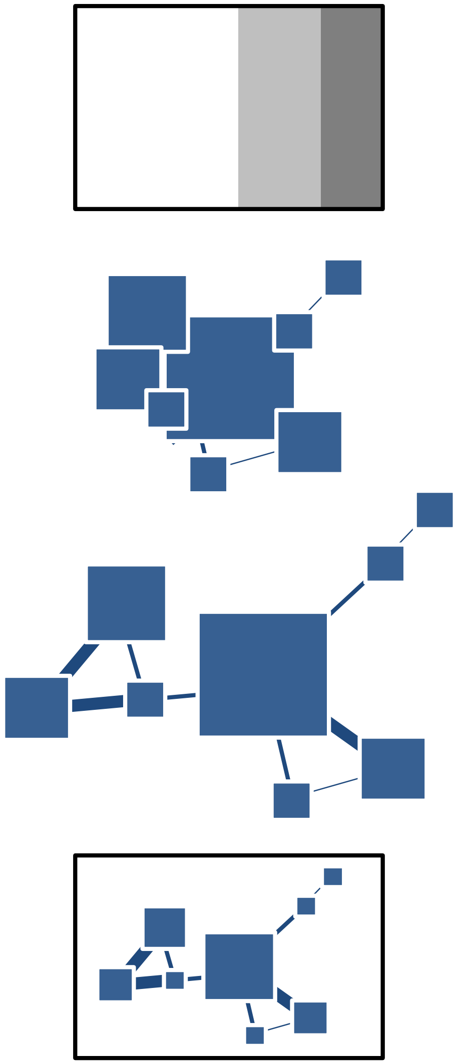
GiB Force-Directed: Algorithm

- Start with initial area usage (20%--50%)
- Generate initial positions
  - Harel & Koren, 2002
  - Better to use meta-edge weights
- Remove overlaps
  - Gansner & Hu, 2009
    - Minimize space used
    - Retain layout structure
- Scale the new layout to fit

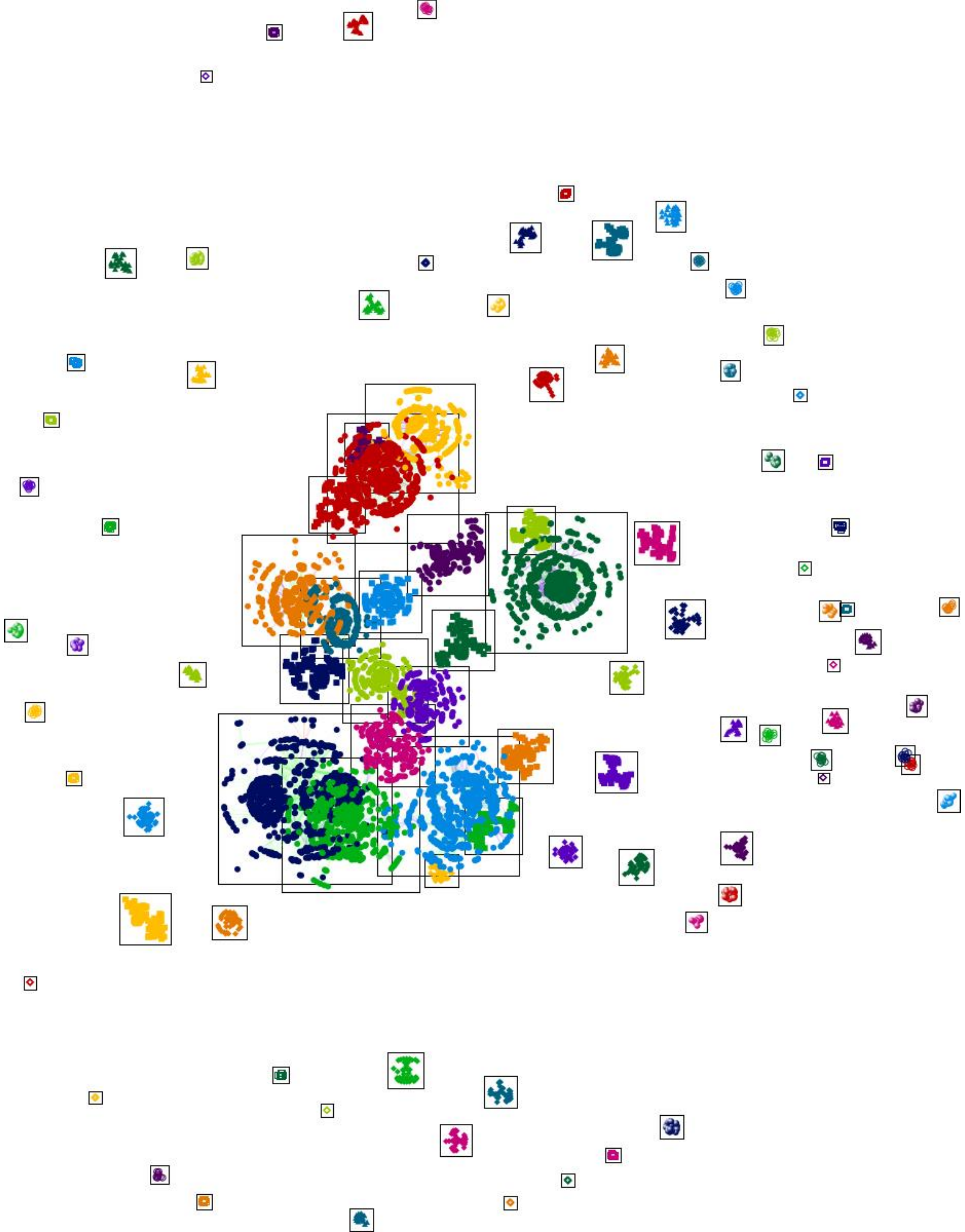


GiB Force-Directed: Algorithm

- Start with initial area usage (20%--50%)
- Generate initial positions
  - Harel & Koren, 2002
  - Better to use meta-edge weights
- Remove overlaps
  - Gansner & Hu, 2009
    - Minimize space used
    - Retain layout structure
- Scale the new layout to fit

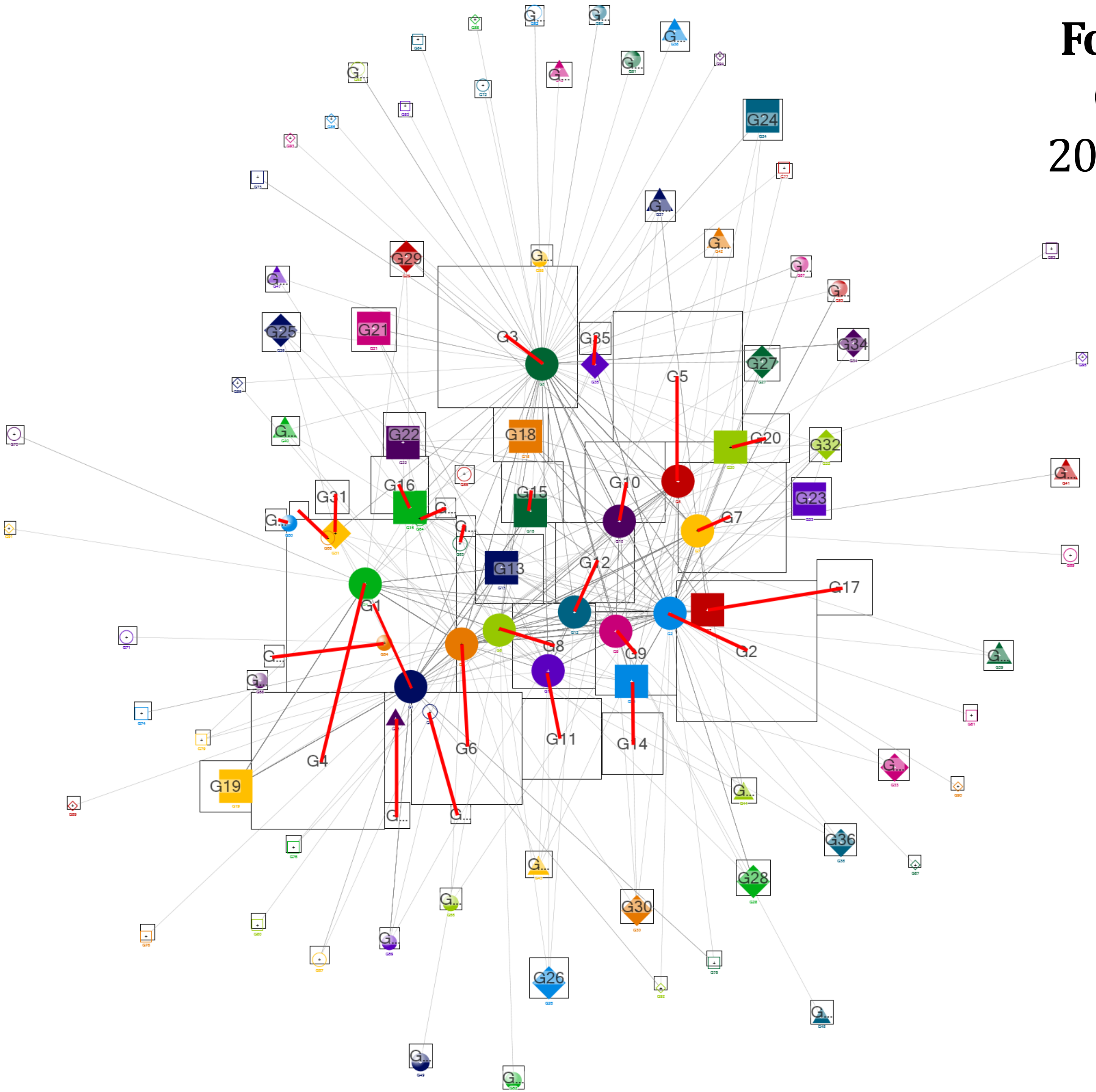


Force-Directed GiB Box Initial Positions



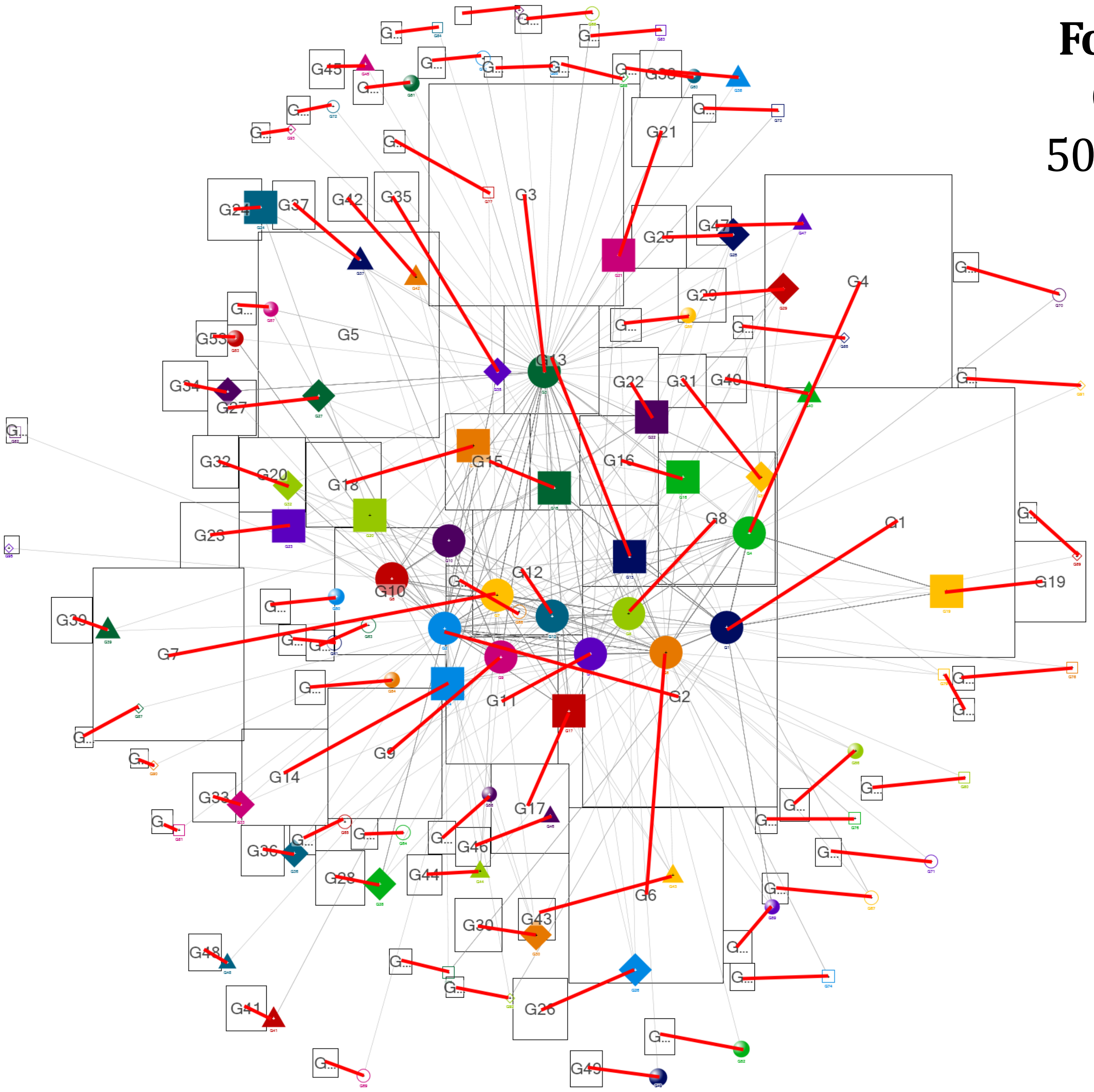
Force-Directed GiB Overlap Removal

20% Originally Filled



Force-Directed GiB Overlap Removal

50% Originally Filled



Putting It All Together

Layout depends on task requirements: space-filling vs. showing relationships

- Treemap
- Croissant-Doughnut
- Force-directed

Automatic choices:

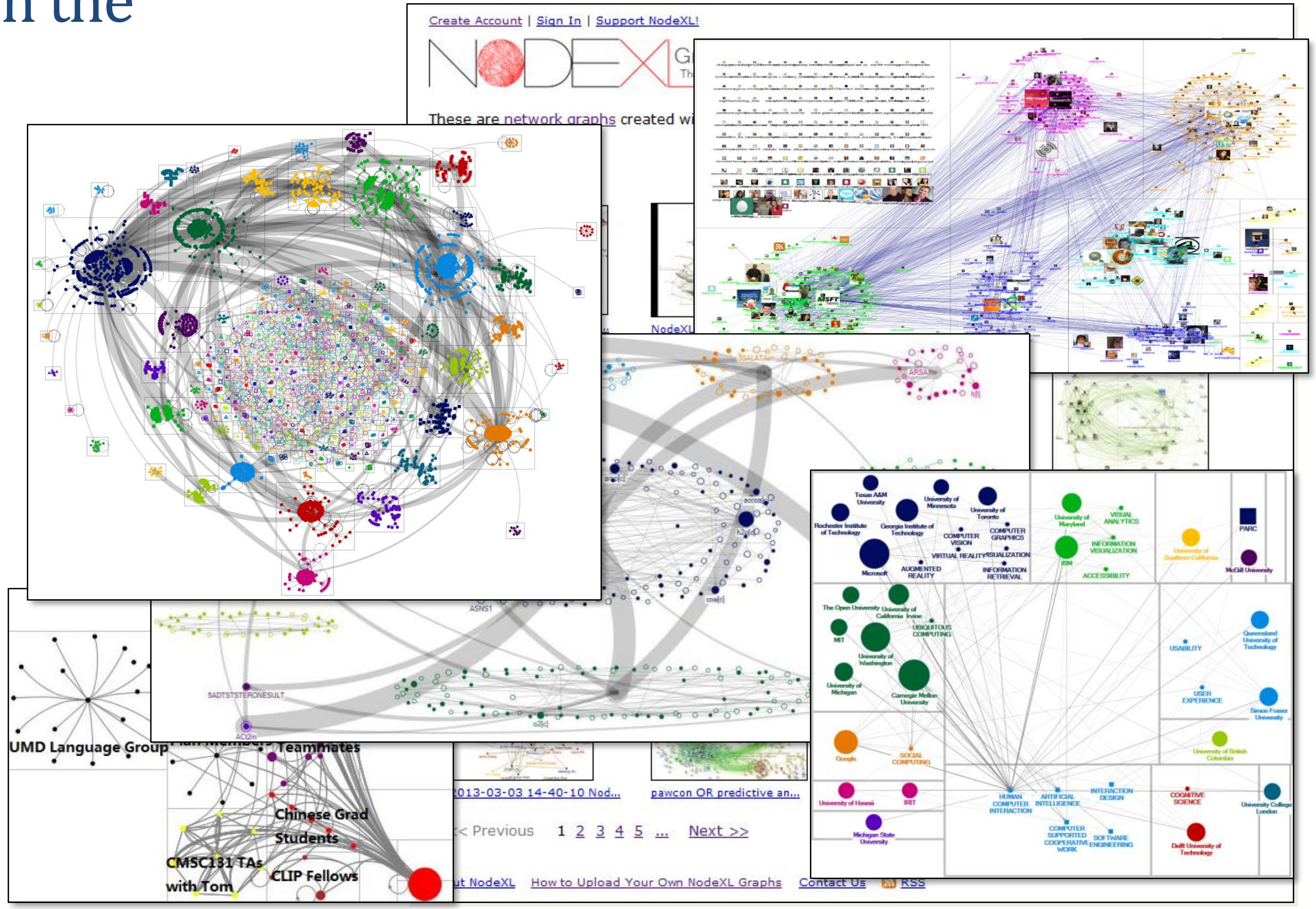
- Croissant-Doughnut
  - Croissant for more evenly distributed groups
  - Doughnut for a few large groups
- Disconnected components*
  - Treemap outer layout
  - Nested GIB layouts
- Rotate/flip to reduce edge crossings*

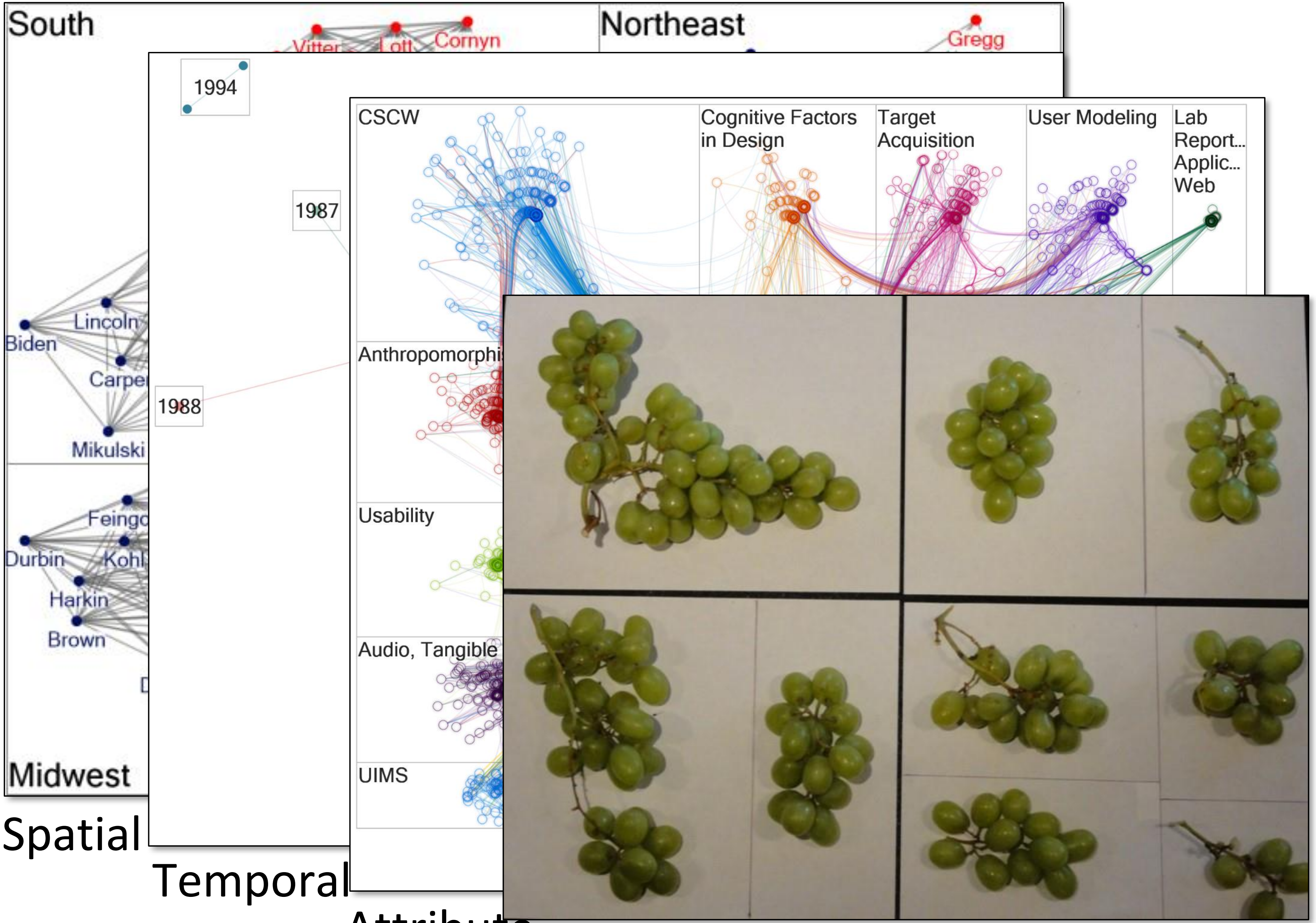
Empirical Evaluation

- Compare techniques on 309 real Twitter networks
- Measure readability issues and time taken (medians shown)
- CD chooses Croissant or Doughnut correctly

				CD-GIB Experiments	
Property/Measure	**ST-GIB**	**CD-GIB**	**FD-GIB**	**Doughnut always**	**Croissant always**
Edge-Box-Overlap (x10^-2)	5.4	5.1	**1.8**	5.4	5.3
Percent Screen Space Blank	**0.0**	2.0	58.7	17.5	2.0
Execution time (in ms)	811.0	**744.0**	951.0	765.0	739.0
Avg Group-Box Aspect Ratio	1.1	2.1	**1.0**	3.5	2.0

Used in the Wild





Spatial

Temporal

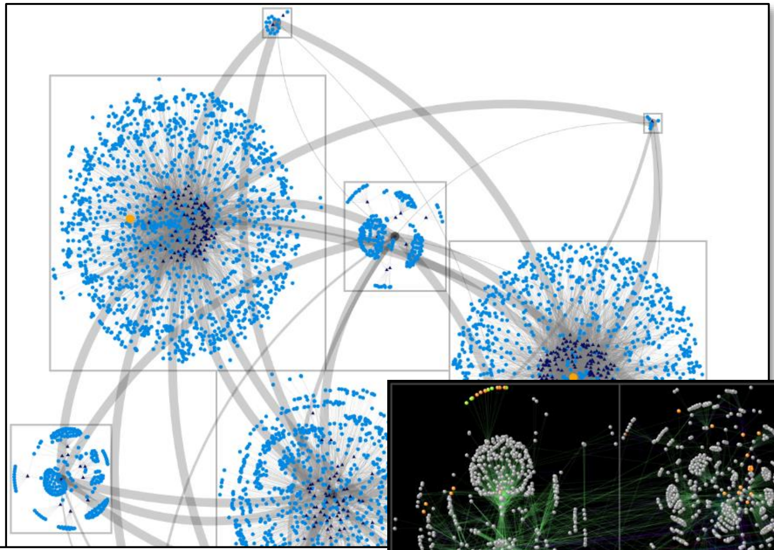
Attribute

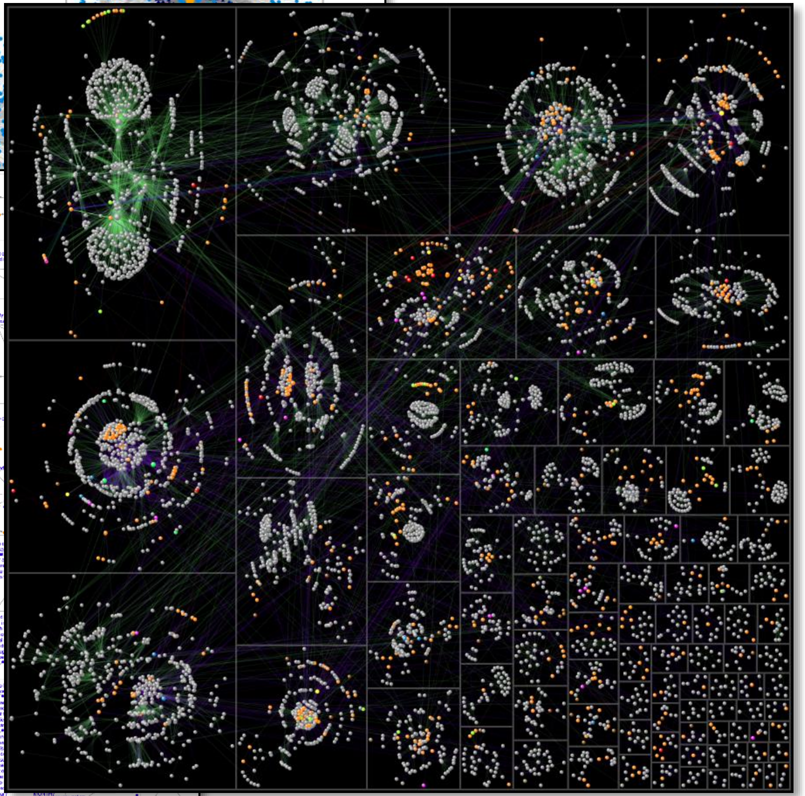
Grapes-in-a-Box

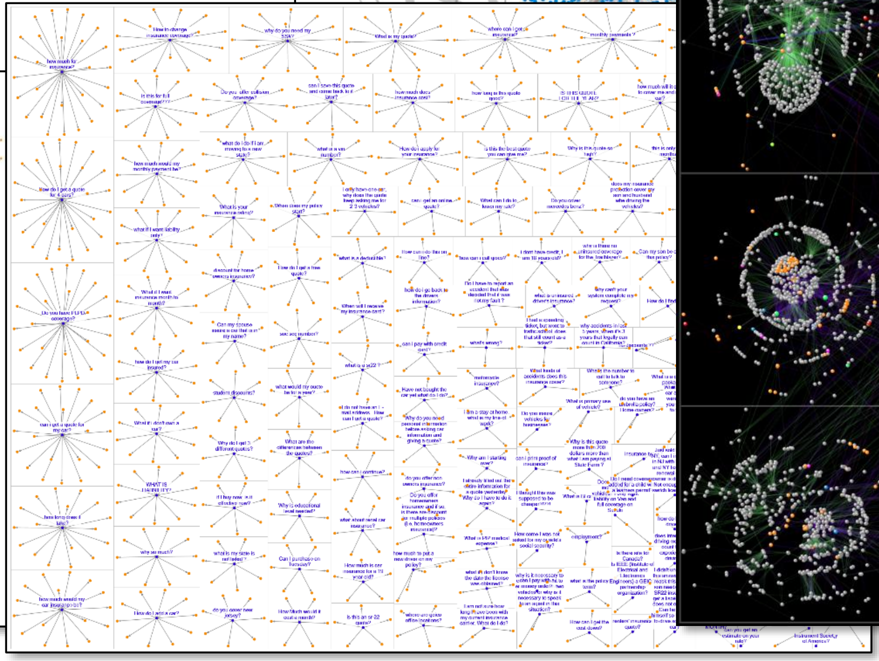
Group-in-a-Box Meta-Layouts

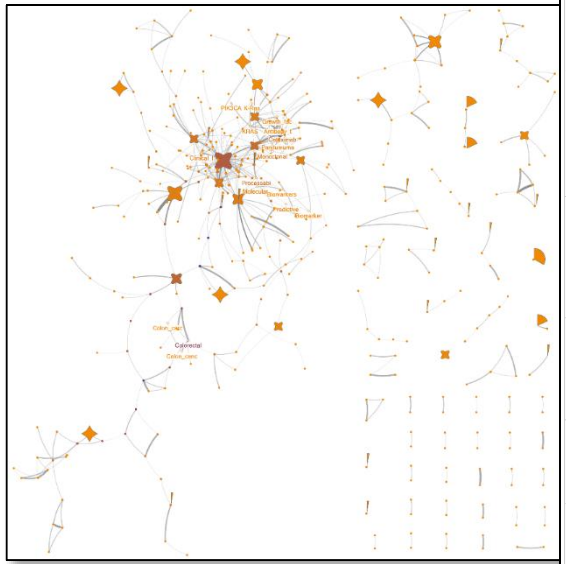
See local and global connections

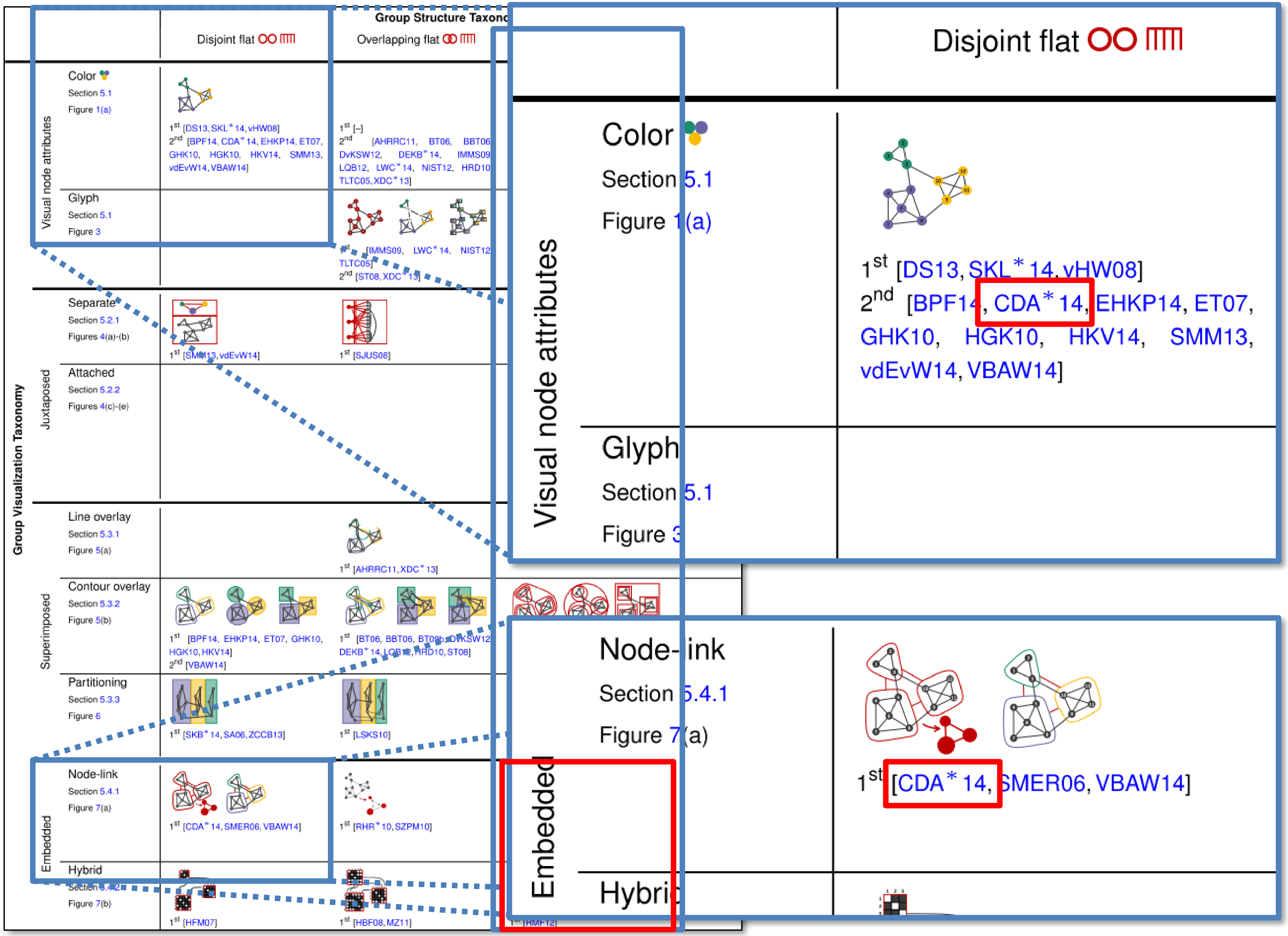
- Concept Insights Biomarker Analysis
- SharpC Medical record concepts
- Similarity for Active Learning
- Innovation in Pennsylvania
- U.S. Senate Voting Patterns



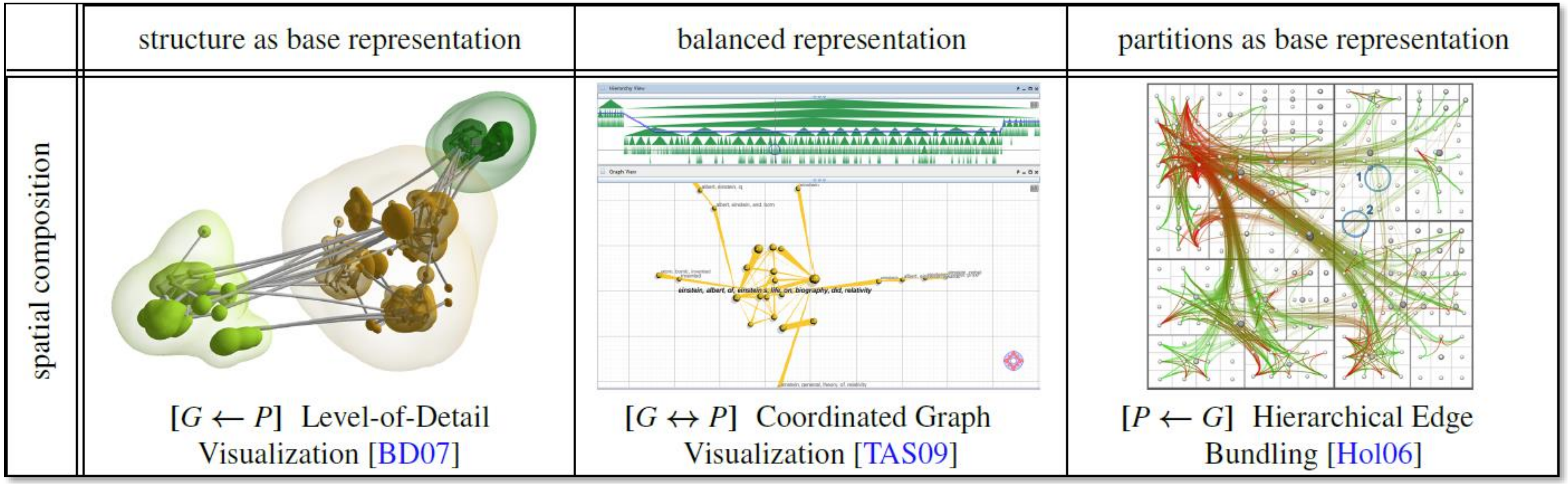




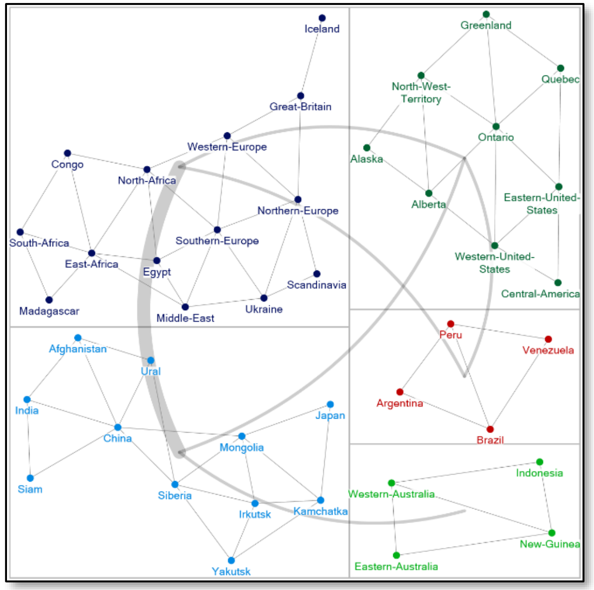


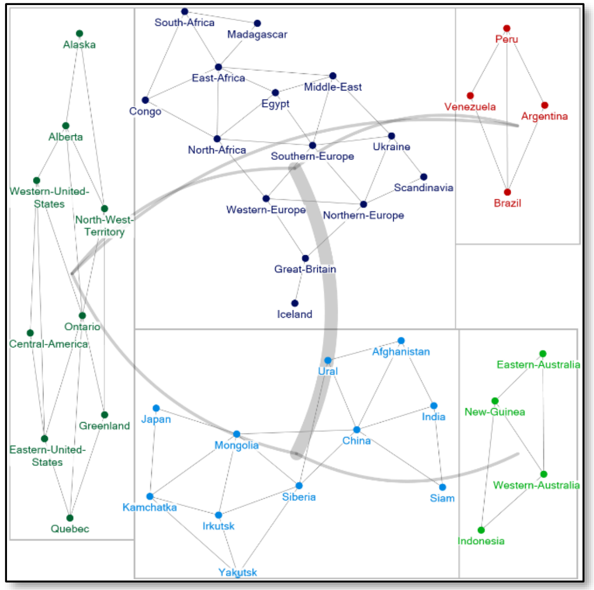


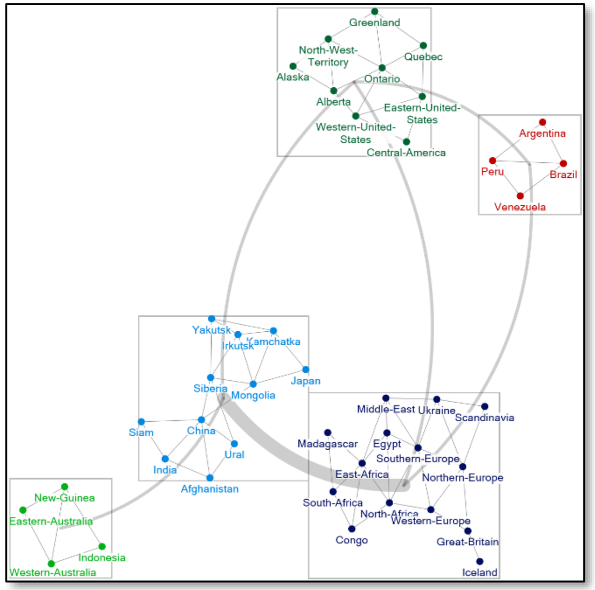
Corinna Vehlow, Fabian Beck, & Daniel Weiskopf (2015)



Steffen Hadlak, Heidrun Schumann, & Hans-Joerg Schulz (2015)





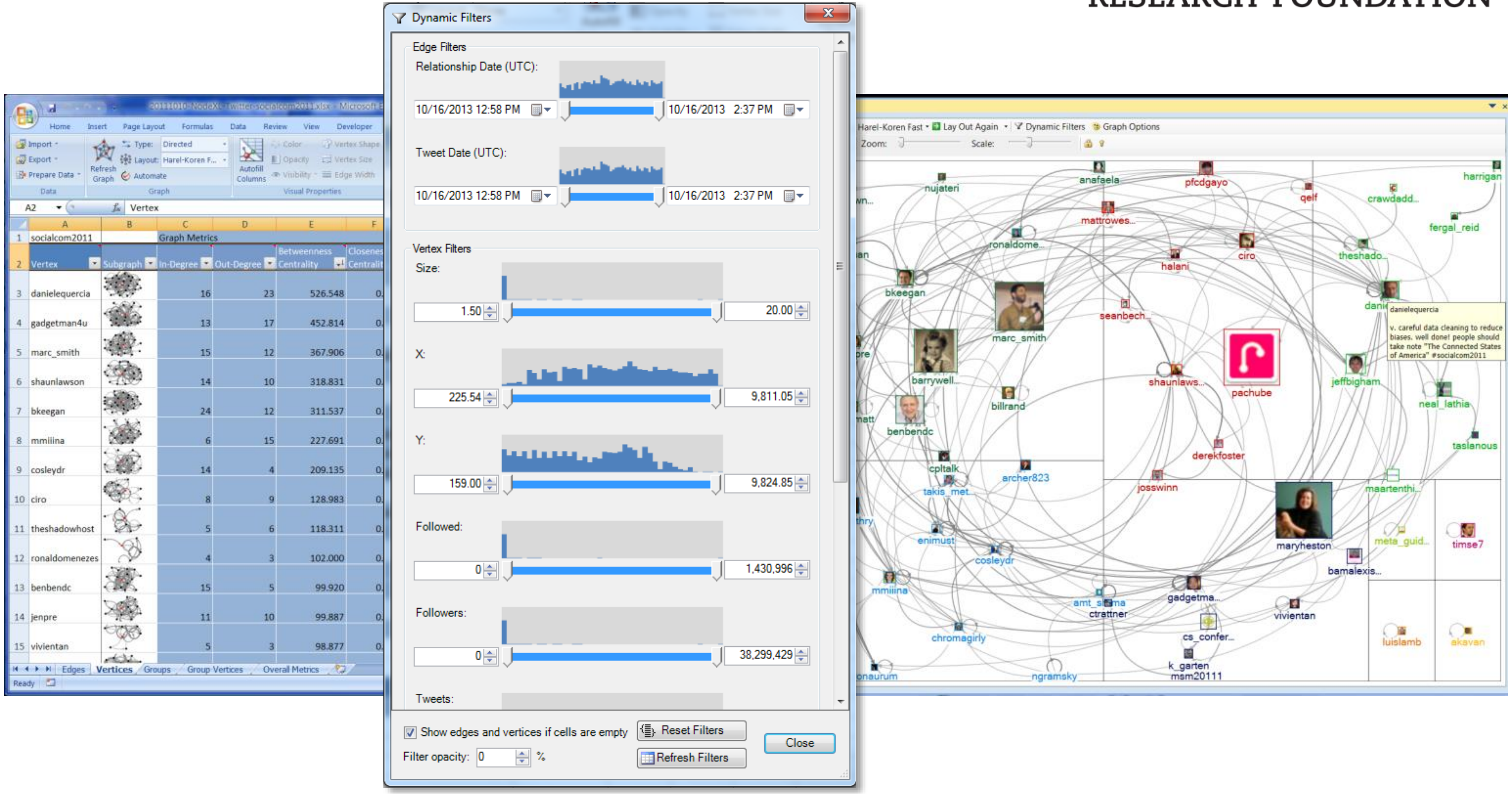


[P ← G] ST-GIB

[G ← [P ← G]] CD/FD-GIB

NODEXL

socialmedia
RESEARCH FOUNDATION



Group-in-a-Box Meta-Layouts

Discussion

- Three Group-in-a-Box **layout algorithms** for dissecting networks
  - Improved group and overview visualization
- **Empirical evaluation** on 309 Twitter networks using readability metrics
- Publicly available **implementation in NodeXL**: nodexl.codeplex.com

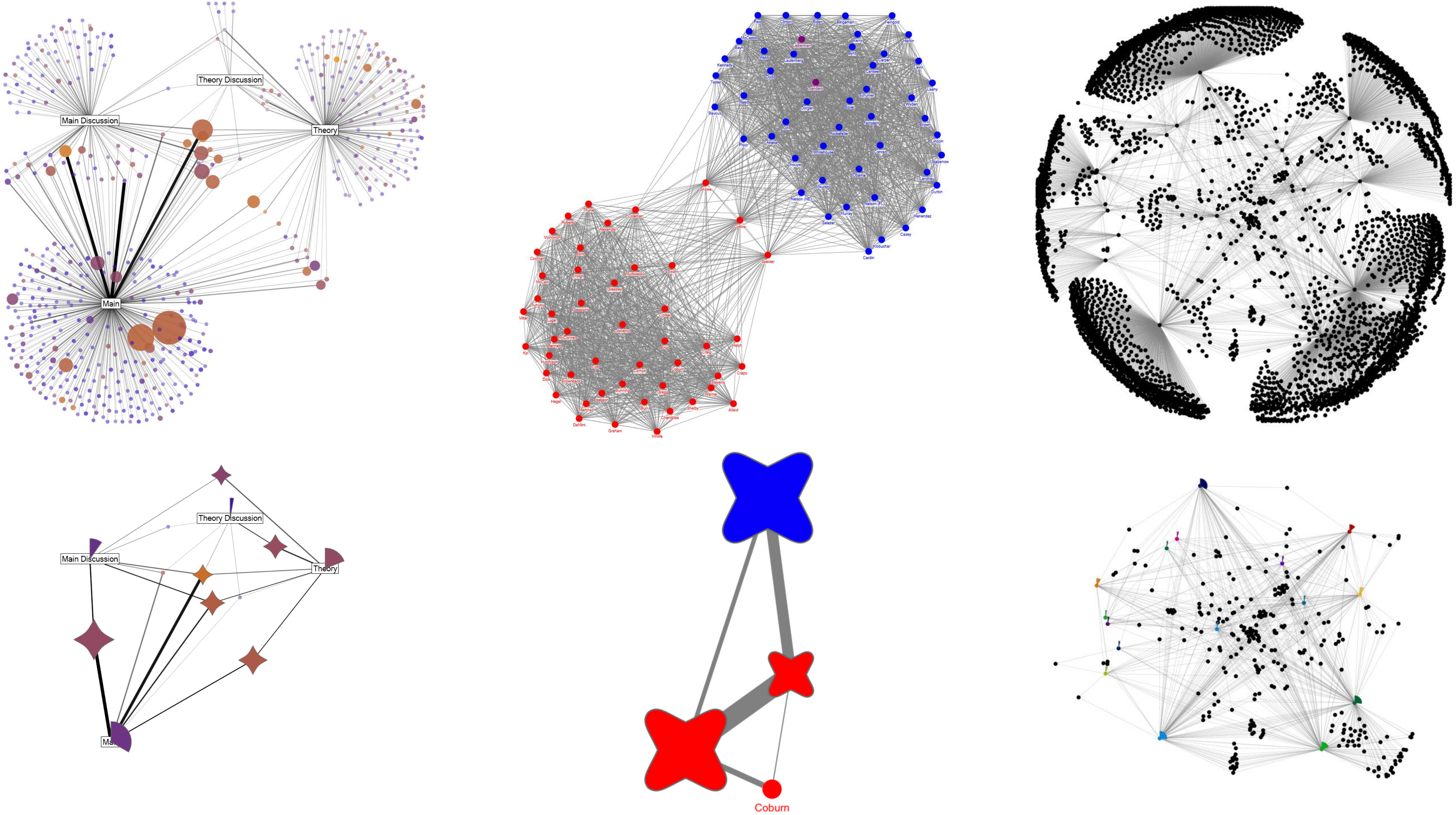
Chaturvedi S, Dunne C, Ashktorab Z, Zacharia R, and Shneiderman B (2014), *"Group-in-a-Box meta-layouts for topological clusters and attribute-based groups: space efficient visualizations of network communities and their ties"*, CGF: Computer Graphics Forum.

Shneiderman B and Dunne C (2012), *"Interactive network exploration to derive insights: Filtering, clustering, grouping, and simplification"*, In Graph Drawing '12. pp. 2-18. DOI:10.1007/978-3-642-36763-2_2

Rodrigues EM, Milic-Frayling N, Smith M, Shneiderman B, and Hansen (2011), *"Group-in-a-Box layout for multi-faceted analysis of communities"*, In SocialCom '11. pp. 354-361. DOI:10.1109/PASSAT/SocialCom.2011.139

Topology Aggregation

Wiki Senate Web Crawl



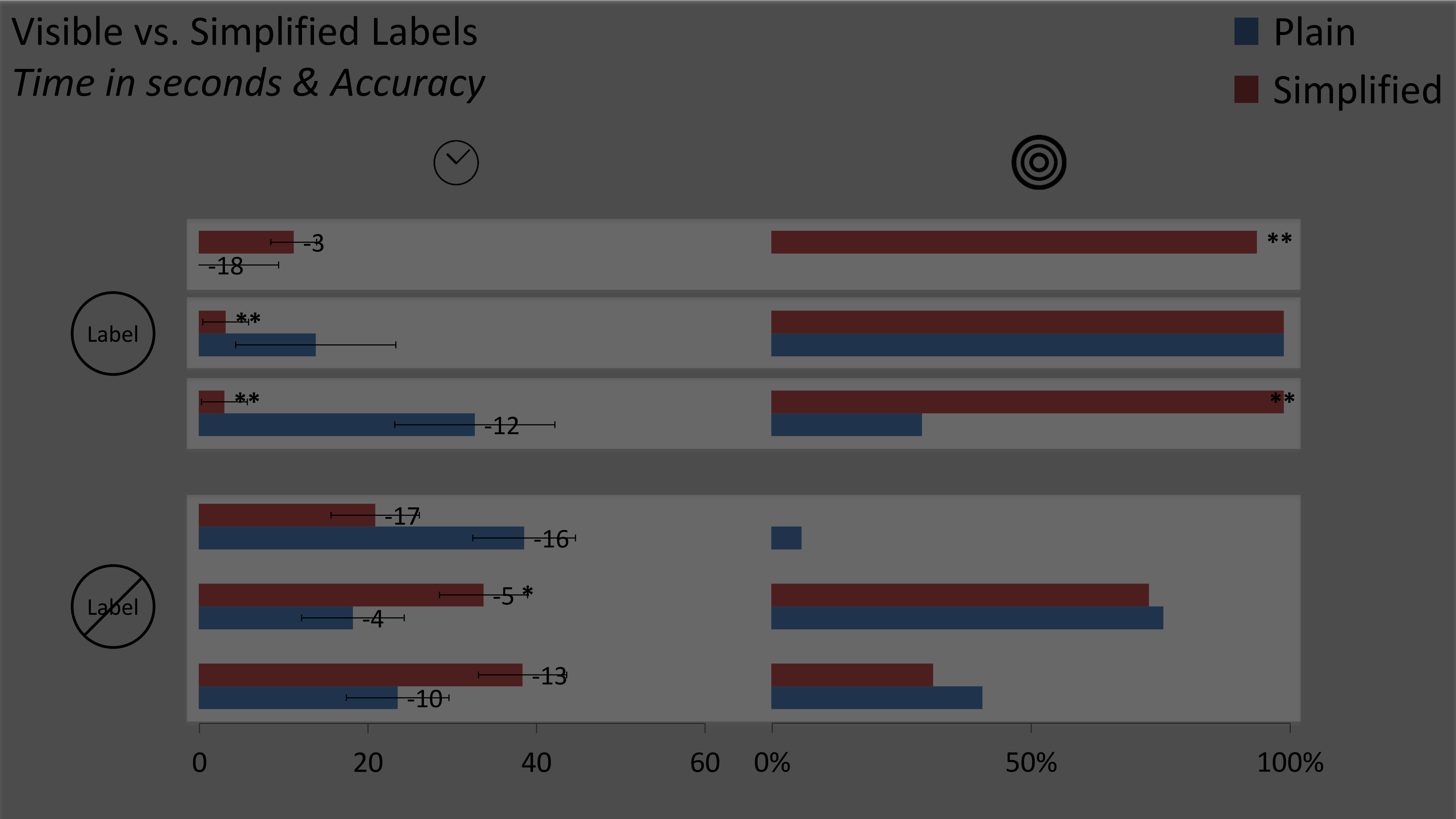
Controlled Experiment

- Participants: 2 pilot, 36 main
- Data: The Wiki, Senate, and Web networks
- Two groups: control and motif simplification
- 31 questions
- 45 minutes

Controlled Experiment - Tasks

Based on Lee et al. 2006 taxonomy:

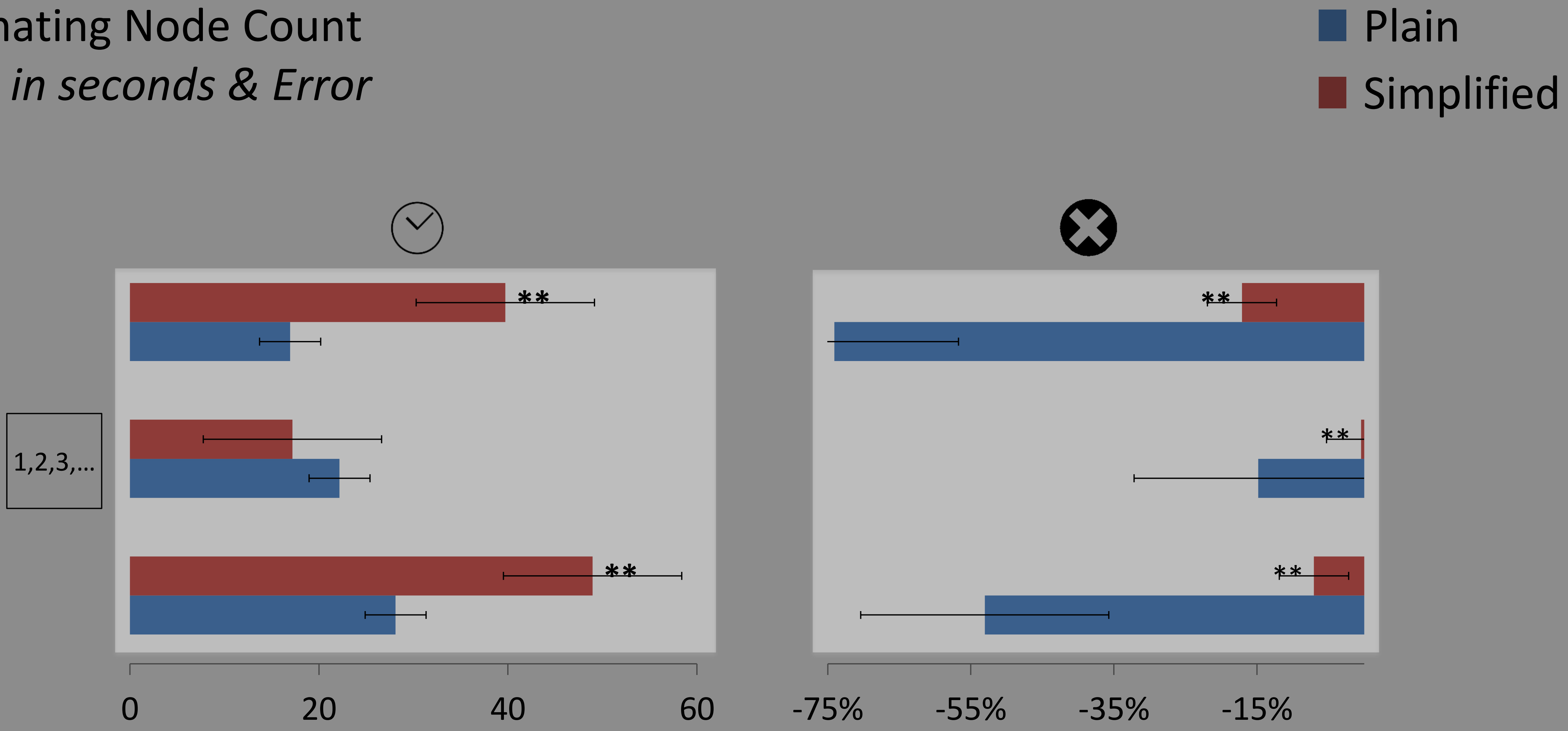
1. Node count: About how many nodes are in the network?
2. Cut point: Which individual node would we remove to disconnect the most nodes from the main network?
3. Largest motif & size: Which is the largest (fan | connector | clique) motif and how many nodes does it contain?
4. Labels: Which node has the label “XXX”?
5. Shortest path: What is the length of the shortest path between the two highlighted nodes?
6. Neighbors: Which of the two highlighted nodes has more neighbors?
7. Common Neighbors: How many common neighbors are shared by the two highlighted nodes?
8. Common Neighbors: Which of these two pairs of nodes has more common neighbors?

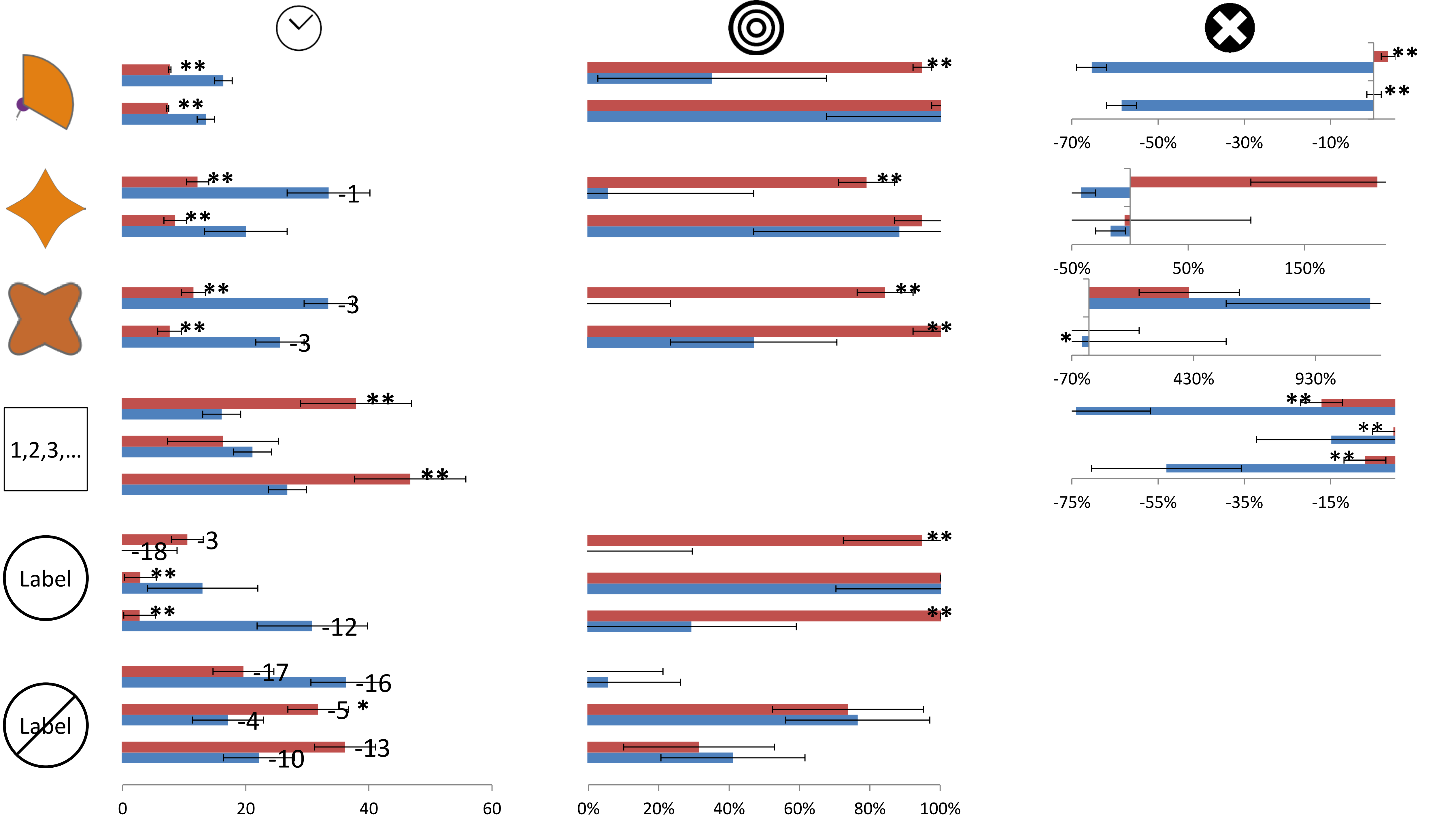


Visible vs. Simplified Labels
Time in seconds & Accuracy
Plain
Simplified
Label
Label
-3
-18
**
**
-12
**
**
-17
-16
-5 *
-4
-13
-10
0
20
40
60
0%
50%
100%

Estimating Node Count

Time in seconds & Error





1,2,3,...
Label
Label
0
20
40
60
0%
20%
40%
60%
80%
100%
-70%
-50%
-30%
-10%
50%
150%
430%
930%
-75%
-55%
-35%
-15%

Motif Simplification

- **Algorithms** for detecting fans, connectors, and cliques
- Publicly available **implementation in NodeXL**: nodexl.codeplex.com
- **Case studies** in political science, sociology, urban planning, medical informatics, intelligence analysis…
- **Controlled experiment** with 36 users showed that motif simplification improves user task performance

Dunne C and Shneiderman B (2013), *"Motif simplification: improving network visualization readability with fan, connector, and clique glyphs"*, In CHI `13.

Shneiderman B and Dunne C (2012), *"Interactive network exploration to derive insights: Filtering, clustering, grouping, and simplification"*, In Graph Drawing `12.

Dunne C, Shneiderman B and Johnson T (2014), *"Understanding patterns in patient discharge summaries using network analysis"*. University of Maryland. Human-Computer Interaction Lab Tech Report No. (HCIL-2014-06).