A similarity-based approach to generate edge bundles

Fábio Henrique Gomes Sikansi

Fábio Henrique Gomes Sikansi

A similarity-based approach to generate edge bundles

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. FINAL VERSION

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Fernando Vieira Paulovich

USP – São Carlos
February 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

Sikansi, Fábio Henrique Gomes
S579a A similarity-based approach to generate edge bundles / Fábio Henrique Gomes Sikansi; orientador Fernando Vieira Paulovich. – São Carlos – SP, 2017.
93 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2017.

1. Edge bundling. 2. graph visualization.
3. information visualization. I. Paulovich, Fernando Vieira, orient. II. Título.

Fábio Henrique Gomes Sikansi

Uma abordagem baseada em similaridade para a construção de agrupamentos visuais de arestas

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. VERSÃO REVISADA

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Fernando Vieira Paulovich

USP – São Carlos
Fevereiro de 2017

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Fernando Paulovich, pela dedicação e comprometimento com este trabalho e com a minha formação, por ter dado liberdade para que eu aplicasse as minhas ideias durante a pesquisa, ao mesmo tempo em que apresentou importantes intervenções que foram determinantes para o resultado final obtido.

Aos muitos colegas que passaram pelo laboratório de Visualização, Imagens e Computação Gráfica do ICMC/USP durante o período de pesquisa. Em especial ao Felipe, Francisco, Samuel, Jorge, Danilo e Tácito, por entenderem que o trabalho na pós-graduação não se restringe ao seu próprio projeto de pesquisa, mas também por olhar ao redor para conversar, trocar experiências e mostrar novas ideias. Essas ações tiveram muito impacto no meu aprendizado e no resultado deste trabalho.

Ao Instituto de Ciências Matemáticas e de Computação e à Universidade de São Paulo, por me moldarem como profissional e como pesquisador, sendo praticamente minha casa durante os últimos 8 anos.

As agências de fomento CAPES e FAPESP, pelo auxílio financeiro para execução deste trabalho.

Ao meu avô José Carlos (*in memoriam*), por ter sido o exemplo de pessoa que eu busco ser. As minhas avós Maria Antônia e Maria Helena, minha mãe Sandra, meu pai Eduardo, minha tia Mônica, pelo incentivo, liberdade e apoio incondicional para que eu realizasse tudo que planejei para minha vida.

À minha namorada Vanessa, pelo amor, apoio e paciência que sempre teve para aguentar os momentos mais estressantes que a pós-graduação pode causar.

This research was supported by FAPESP (grant number 2014/18665-1) and CAPES. The opinions, assumptions, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of FAPESP and CAPES.

ABSTRACT

SIKANSI, F.. A similarity-based approach to generate edge bundles. 2017. 93 f. Master dissertation (Master student Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Graphs have been successfully employed in a variety of problems and applications, being the object of study in modeling, analysis and construction of visual representations. While different approaches exist for graph visualization, most of them suffer from the severe clutter when the number of nodes or edges is large. Among the approaches that handle such problem, edge bundling techniques attained relative success on improving the quality of the visual representations by bending and aggregating edges in order to produce an organized layout. Despite this success, most of the exiting techniques create edge bundles based only on the visual space information, that is, there is no explicit connection between the edge bundling layout and the original data. Therefore, these techniques generates less meaningful bundles and may lead users to misinterpret the data. This master's research presents a novel edge bundling technique based on the similarity relationships among vertices. We developed such technique based on two assumptions. First, it supports the hypothesis that edge bundling can better represent the data when there is an inherent connection between the proximity among the elements in the information space and the proximity between edges in the edge bundling layout. We address this question by presenting a similarity bundling framework, that considers the similarity between vertices when performing the edges bending. To guide the bundling, we create a similarity hierarchy, called backbone. This is based on a multilevel partition of the data, which groups edges of similar vertices. Second, we also support that a multiscale representation improves the visual and complexity scalability of bundling layouts. We present a multiscale edge bundling, which allows an overview plus detailed exploration, coarsening or revealing the bundling at different levels of the same visualization. Our evaluation framework shows that our backbone produces a balanced hierarchy with a good representation of similarity relationships among vertices. Moreover, the edge bundling layout guided by the backbone reduces the visual clutter and surpass state-of-the-art techniques in displaying global and local edge patterns.

Key-words: Edge bundling, graph visualization, information visualization.

RESUMO

SIKANSI, F.. A similarity-based approach to generate edge bundles. 2017. 93 f. Master dissertation (Master student Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Grafos são empregados com sucesso em uma grande variedade de problemas e aplicações, sendo objeto de estudo na modelagem, análise e na construção de representações visuais. Embora existam diferentes formas para a visualização de grafos, a maioria delas sofrem pela desorganização do espaço visual quando o número de vértices ou arestas é alto. Entre as abordagens que lidam com este problema, as técnicas de agrupamentos visuais de arestas obtiveram sucesso na melhora da representação visual pelo encurvamento e agrupamento de arestas que aperfeiçoam a organização da representação. Apesar deste sucesso, a maioria das técniques criam grupos de arestas baseados apenas na informação do espaço visual, não existindo conexão explícita entre o desenho no espaço visual e o conjunto de dados original. Dessa forma, estas técnicas produzem agrupamentos de arestas com baixa significância e podem levar o usuário a uma interpretação incorreta da informação. Esta pesquisa de mestrado apresenta uma nova técnica de agrupamento visual de arestas baseado nas relações de similaridade entre os vértices. Nós desenvolvemos esta técnica com base em duas premissas. Primeiro, ela defende a hipótese que a representação por agrupamento de arestas pode representar melhor o conjunto de dados se existir uma conexão inerente entre a proximidade dos elementos no espaço de informação e a proximidade entre arestas no desenho de arestas agrupadas. Nós atendemos esta questão apresentando um arcabouço para o agrupamento de arestas baseado em similaridade, que considera a similaridade entre vértices para realizar o encurvamento das arestas. Para guiar este encurvamento, nós criamos uma estrutura de similaridade, denominada backbone. Esta estrutura é baseada em um particionamento multi-nível do conjunto de dados, que agrupa arestas de vértices similares. A segunda premissa, nós também defendemos que uma representação multiescala melhora a escalabilidade computacional e visual da representação visual de arestas agrupadas. Nós apresentamos um agrupamento visual multi-nível de arestas que permite uma exploração generalizada e detalhada, revelando detalhes em múltiplos níveis da visualização. Nosso processo de avaliação mostra que a construção do backbone produz uma hierarquia balanceada e com boa representação das relações de similaridade entre os vértices. Além disso, a visualização com arestas guiadas pelo backbone reduz a desordem visual e melhora as técnicas do estado-da-arte na identificação de padrões de arestas globais e locais.

Palavras-chave: Agrupamento visual de arestas, visualização de grafos, visualização de informação.

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

SDEB .... Similarity-driven Edge Bundling
ADEB .... Attribute-Driven Edge Bundling
CUBu .... CUDA-based Universal Bundling
CUDA ... Compute Unified Device Architecture
DEB ..... Divided Edge Bundling
FDEB .... Force-Directed Edge Bundling
GBEC .... Geometry-based Edge Clustering
GPU ..... Graphics Processing Unit
HEB ..... Hierarchical Edge Bundling
IBEB ..... Image Based Edge Bundles
KDEEB .. Kernel Density Estimation Edge Bundling
MINGLE . Multilevel Agglomerative Edge Bundling
NJ ..... Neighbor-Joining
SBEB .... Skeleton-Based Edge Bundling
STree .... Similarity Tree
UPGMA .. Unweighted-pair Group Method with Arithmetic Means
WR ..... Winding Roads

CONTENTS

CHAPTER

1

INTRODUCTION

A graph is commonly defined as a set of vertices (also called nodes) and edges that represent relationships between vertices. This is considered one of the most important information structure in discrete mathematics and computer science. The fundamentals, concepts, topology and geometry of graphs are part of the graph theory area. Graph theory started with the famous Königsberg Bridge Problem, which consisted of determining if it is possible to walk by a set of seven bridges over the river Preger in Königsberg without crossing any bridge more than once. This problem was modeled as a graph and answered as negative by Euler in 1735 (ALEXANDERSON, 2006). More than 270 years after, graph theory is still an evolving field used to model and solve problems.

One of the most important characteristics of graphs is the possibility of mathematically model diverse real-world scenarios and multiple tasks. Examples of graph theory usages exist in economics (SERRANO; BOGUÑÁ, 2003), sociology (MERCKEN et al., 2010), biology (KIKUCHI et al., 2003), geography (NOCAJ; BRANDES, 2013), physics (DOYE; MASSEN, 2005), transport engineering (BEN-AKIVA; PALMA; ISAM, 1991) and software engineering (ELZEN et al., 2013).

Recent events quickly changed the way that users analyse data. The growing amount of information brought an extensive collection of challenges on the storage, analysis and exploration of information (KEIM, 2002). We live in the Information Era, in which data has been generated faster and in a greater volume than the analytical capacity of an average human can handle. Examples of the availability of information include reaching a person among billions in a global coverage social network like Twitter[1] or Facebook[2]; tracking every flight in the Earth with Flight Radar[3]; or reading reviews of hotels, restaurants and attractions before planning a trip on

[1] https://twitter.com/
[2] https://www.facebook.com/
[3] http://www.flightradar24.com/

TripAdvisor[4].

As great as the challenges it represents, that amount of information can produce the necessary knowledge to detect and understand behaviors, solve problems and make decisions that improve several aspects of human life. Nowadays, data comes from a wide range of sources and directions, and new models and processes are crucial to deal with that information. In computer science, many research topics are dedicated to address these problems, such as data mining, machine learning, visualization and visual analytics.

Purely computational techniques might replace the human activity in order to solve many problems, mainly when the problems can be strictly formulated. However, there are still some tasks that request user interaction. The process of visual representation of the data and the interaction with and a human is studied by the field called visualization (MUNZNER, 2014). Specifically, information visualization is the representation of data in a visual space, by drawing methods that form an image to improve the ability of a user to interpret the data (SPENCER, 2007). The visualization pipeline involves concepts from different areas, such as mathematics, computering, perception and cognitive sciences (TELEA, 2007).

The crossroad of graph theory and information visualization is the field of graph drawing. Drawing a graph is the method of building a visual representation of vertices and edges. There are a variety of algorithms that perform this task, often taking advantage of graph properties like planarity, symmetry, edge direction and cycles (BATTISTA et al., 1998). Techniques may also leverage the underlying data to produce better layouts (LANDESBERGER et al., 2011; BECK et al., 2014). For complex inputs with many elements and edge crossings, the drawing method can fail to produce a valuable visual representation, thus being an open challenge on information visualization. More precisely, a problem frequently faced in graph visualization is the visual clutter. In essence, visual clutter is the reduction of the usefulness of a visualization by the excessive number of elements in a limited visual space (ELLIS; DIX, 2007).

In graph visualization, the visual clutter problem has been often addressed by the reorganization, reduction and aggregation of elements (EPPSTEIN; GOODRICH; MENG, 2007; JIA et al., 2008; HENRY; BEZERIANOS; FEKETE, 2008; DWYER et al., 2013), or modifying how the information is displayed (CUI; QU, 2007). This can be achieved by changing the vertices and edges placement. Among those strategies, edge bundling techniques aim to reduce the clutter by drawing curved and aggregated edges through similar roads. In this process, close edges share routes and are drawn together, reducing the visual space usage and improving the usefulness of the visualization by representing edge patterns and groups.

Edge bundling was introduced by the technique Hierarchical Edge Bundling (HEB) (HOLTEN, 2006) in a process that involved the usage of a hierarchy to determine a set of paths and control points. Once this hierarchy connects all graph vertices through control points, the

[4] https://www.tripadvisor.com.br/

original edges are replaced by B-Splines curves, which follow the path that connects the source and the target over the hierarchy. HEB produces more pleasant layouts and its effectiveness was showed by the visualization of software dependencies and traits (HOLTEN; CORNELISSEN; WIJK, 2007). However, the hierarchy dependence does not allow this technique to be applicable to a large range of datasets.

Subsequently, several other approaches have been developed to perform edge bundling without an external structure to guide the edge bending and to process larger datasets. These methods often take advantage of geometry information, parallel and GPU computing. For those techniques, we use a taxonomy that considers the main concepts under its layout construction process. This taxonomy divides the techniques in geometry-based approaches (CUI et al., 2008; LAMBERT; BOURQUI; AUBER, 2010), force-based approaches (HOLTEN; WIJK, 2009), image-based and GPU techniques (TELEA; ERSOY, 2010; ERSOY et al., 2011; HURTER; ERSOY; TELEA, 2012; ZWAN; CODREANU; TELEA, 2016). The evolution of edge bundling techniques allowed the bundling of graphs with thousands or even millions of edges to be processed in an order of hundredths of a second, and to deal with real-time dynamic graphs (NGUYEN; EADES; HONG, 2013b; HURTER et al., 2013).

However, all those techniques mainly use the spatially information to perform the edge bundling and ignore the underlying data, thereby creating less meaningful aggregations that might not explicitly reflect the data. Recently, the use of data information to approximate the edges gained attention in some publications (NGUYEN; HONG; EADES, 2012; PEYSAKHOVICH; HURTER; TELEA, 2015; GUO et al., 2015; YAMASHITA; SAGA, 2015; SUN et al., 2016). They modified prior techniques to use related attributes when bending edges. However, these changes only performed minor transformations in the edge bundling layout.

This master's research defends that similarity based approaches to build a bundling-oriented hierarchy can increase the readability of edge bundling layouts. This is achieved by taking an exclusive data-oriented approach that determines how the edges share their routes, such as the hierarchy in the Hierarchical Edge Bundling. This method has a straightforward methodology and can easily scale when the number of edges increase. Although there are many techniques for hierarchical construction from similarity information, to the best of our knowledge there are no studies regarding the application of those approaches on the Hierarchical Edge Bundling.

During this master's research, we analyzed some algorithms to build hierarchical layouts from a set of elements, such as the Neighbor-Joining (NJ) (SAITOU; NEI, 1987) and the Unweighted-pair Group Method with Arithmetic Means (UPGMA) (SOKAL; MICHENER, 1958). Initially, those algorithms were applied directly to the edge bundling construction. Then, we considered cluster-based (JAIN; DUBES, 1988) and hybrid (clustering/similarity tree construction) algorithms to obtain a more stable and faster method. Our final methodology covers the hierarchical construction, the points placement into the visual space and the edge bending process.

Our technique was then compared with edge bundling algorithms from the state-of-the-art.

The result of this research is a novel edge bundling technique, called Similarity-driven Edge Bundling (SDEB), that produces an edge bundling layout guided by an accurate similarity structure. This structure is built with a hierarchical clustering method that uses only data from the original data vertices. Moreover, our technique is fast enough and offers the possibility of visualizing large graphs using a multiscale approach that filters similar data objects into clusters, thus reducing the visual-clutter.

1.1 Goals, Evaluation and Contributions

Edge bundling state-of-the-art techniques achieved great success on visual-clutter reduction. However, these techniques do not take into account the underlying data when performing edge aggregation and do not validate the usefulness of the produced layout. This research aimed to produce a more faithful bundling layout by using a bundling process that only considers similarities amongst vertices to determine how edges are bundled. The research goal is described formally in the following paragraph:

> "This master's research aimed to develop a new edge bundling technique for graph visualization. This technique must produce edge bundling layouts that reduce the visual clutter on graph visualization and diminish the distortion caused by differences between the representation of aggregated edges and the similarity relationships among vertices. Furthermore, this technique must handle multiple levels of detail and filter the elements on large datasets, where the process of bundling edges is not enough to reduce the visual clutter."

The main result of this research is a technique called Similarity-driven Edge Bundling, which was built from a framework designed to construct edge bundling layouts. This framework uses a similarity-based algorithm to build a backbone from the original set of vertices. Then, the bundling is created by drawing the original edges towards the backbone. Edges are drawn as curves towards the intermediate vertices from the path that connects the edge source and target in the backbone. Edges that share intermediate vertices are rendered together, which reduces the drawing occupation area and, hence, reduces the visual-clutter. Further enhancements give the user parameters that may be used to manipulate the visualization.

Even though the visual-clutter reduction is recognizable, it is difficult to validate edge bundling layouts because there are a few experiments or measures that evaluate the gain in terms of data analysis or pattern recognition. This problem was mentioned before in many publications (ERSOY et al., 2011; NGUYEN; EADES; HONG, 2013a; HURTER et al., 2013). In general, edge bundling techniques do not validate the usefulness of the bundling layout. In order to perform an evaluation of patterns and their relationships with the similarity structure in graph

layouts, we compare our technique with others, covering the major edge bundling strategies in the state-of-the-art.

From our evaluation process, we outline the following contributions of this research:

- A novel edge bundling layout obtained from the similarity hierarchy that improves the visual and computational scalability, which allows the technique to work with bigger datasets and to be explored in a multi-level approach.
- A framework to build edge bundling layouts from similarities of vertices which can meaningfully simplify the visualization by adding meaning to the bundling process.
- Applications of Similarity-driven Edge Bundling to visualize static graphs from different kinds of data, such as paper citations or co-purchased items.
- A straightforward application in a specific type of dynamic graph, in which the set of edges changes over time, but vertices and their similarity relationships are fixed.

1.2 Structure of the document

The remainder of this document is organized as follows:

- Chapter 2 presents and discusses fundamental concepts on graph visualization and the state-of-the-art on edge bundling.
- Chapter 3 describes the similarity bundling framework and methods to improve the bundling layout.
- Chapter 4 reports the main results and the evaluation of our technique.
- Chapter 5 presents the applications of our technique in real-world datasets.
- Chapter 6 discusses the contributions and limitations of this thesis, along with some directions for future work.

CHAPTER

2

BACKGROUND

2.1 Initial Remarks

In this chapter, we present the fundamentals on graph drawing and the literature review of edge bundling. We classify these methods according to their approach to create the edge bundling layout. Prior to the discussion of the general concepts and the edge bundling methods, we give some definitions:

Definition 1. A graph $G = (V;E)$ is a pair of finite sets V and E, in which the elements of E are called edges and the elements of V are called vertices (or nodes). Each edge connects a pair of vertices $(v_i;v_j)|v_{i;j} \in V$ (GROSS; YELLEN, 2005).

Definition 2. A vertex $v_i \in V$ is defined as adjacent to a vertex $v_j \in V$ if the pair $(v_i;v_j) \in E$.

Definition 3. An edge can be an ordered pair, when it represents a directed connection between the source and target vertices, or an unordered pair, when $(v_i;v_j)$ and $(v_j;v_i)$ represent the same connection. A graph of ordered edges is defined as direct graph (or digraph).

Definition 4. A graph is defined as connected if there is at least one path between any pair of vertices $(v_i;v_j) \in V$. In contrast, if there is any pair of vertices $(v_i;v_j)$ breaking this rule, a graph is defined as disconnected.

Definition 5. A graph is defined as cyclic if there is at least one path of vertices that starts and ends in the same vertex. In contrast, a graph is defined as acyclic if there is no path satisfying such condition.

Definition 6. A graph is a tree if it is both connected and acyclic.

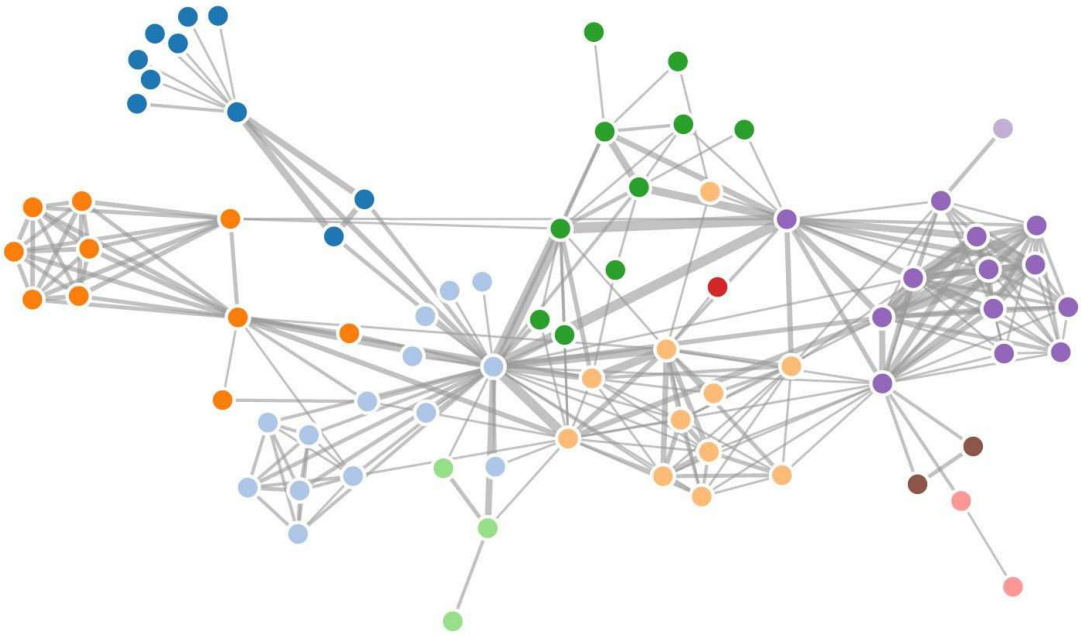
A graph can be used to model instances and the relationship amongst them. In addition, any vertices or edges can contain attributes, such as their size, length, weight or any other related

information (THULASIRAMAN; SWAMY, 2011). In information visualization, we are mainly interested on the visual representation of a given graph. In the next section, we present a review of graph drawing algorithms.

2.2 Graph Visualization

Information visualizations studies how a graph can be represented in a visual space. Practical examples of this task are adjacency matrices and node-link diagrams. To create a node-link diagram, vertices are represented by some entity in the visual space, and edges are represented with lines connecting each pair of vertices. In this case, a good vertices placement is crucial to achieve useful representation. Figure 1 shows a graph visualization using a node-link diagram for a graph with 77 vertices and 254 edges. The vertices are organized in the space by a force-based system. Although this visualization is effective, there is a huge challenge on obtaining a readable and pleasant visualization. Moreover, this challenge is more evident when the number of vertices and/or edges increases.

Figure 1 – A node-link diagram for the undirected graph Les Miserables (KNUTH, 1993) with 77 vertices and 254 edges. The vertices are represented as circles and the edges are lines connecting pairs of vertices.



Source: Adapted from <http://bl.ocks.org/mbostock/4062045>

Several graph drawing algorithms have been published to improve this representation. In this range of algorithms, there are general ones, applied for generic graphs, and others designed for specific cases, often taking advantage of one or more graph properties (e.g., if it is planar, undirected, or a tree) or data information. In this section, we discuss the most common approaches. It is not our intention to present the state-of-the-art on graph visualization. Moreover, we also present some tree drawing algorithms, since this structure is a key concept in this research.

A frequent challenge in graph drawing is the size of the input graph. Graphs with more vertices and edges hamper the object placement on the visual space. Moreover, vertices occlusion and edge crossings reduce the efficacy of the visualization, because it becomes impossible to discern between vertices and edges (HERMAN; MELANÇON; MARSHALL, 2000). The most common way to address this challenge is to improve the elements distribution across the visual space. In general, a graph drawing algorithm must reduce the elements congestion in sub-areas of the drawing space, thus maximizing the visual space occupation (LANDESBERGER et al., 2011). A pleasant layout also has a good aspect ratio and symmetry (WARD; GRINSTEIN; KEIM, 2010). It is desirable that the drawing algorithm always presents a similar output given the same input, in order to prevent the context loss.

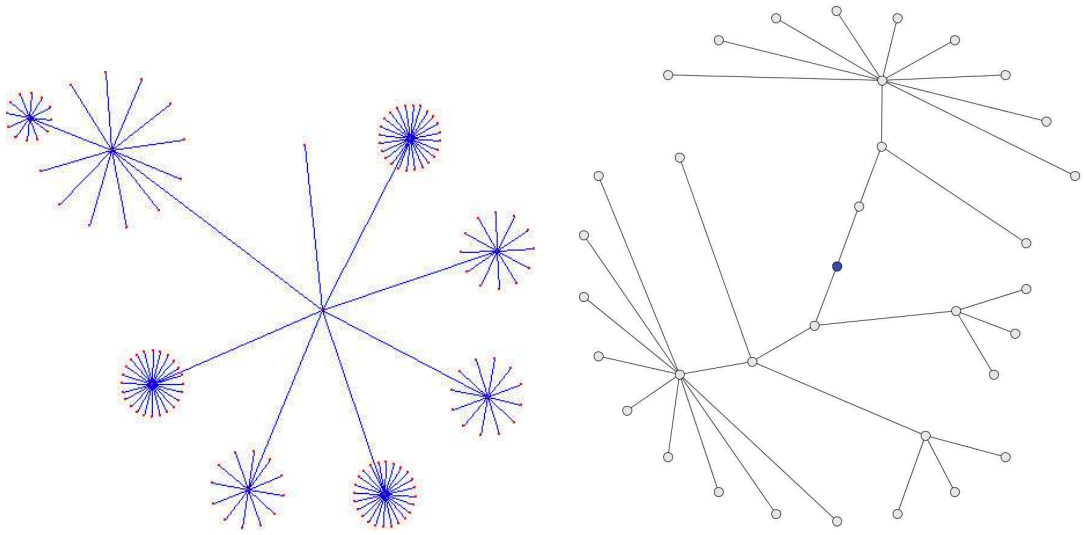
Those rules can be easily described, but it is a tough task to apply them in the drawing process. In detail, it is only possible to avoid edge crossings when the graph is planar, which can be done by planar layout algorithms (KAUFMANN; WAGNER, 2003). Therefore, the goal when we have non-planar graphs is to minimize the number of crossings. When vertices coordinates are fixed according to their attributes (e.g., geography locations), drawing algorithms can curve edges to improve the layout. Other properties, such as symmetry and aspect ratio are susceptible to the inherent information. These characteristics make the usage of a general algorithm difficult for all kinds of graphs.

When the vertices coordinates are not given, an often employed approach in general graphs is the force-based algorithm. This strategy uses a physical simulation of a force system, in which visual elements move in each iteration until the system reaches its stability. In this case, vertices form a system of particles. A force of attraction is applied between adjacent vertices and a repulsive force separates close vertices. According to the forces, the final result might depend on the initialization placement, and other constraints can be applied, such as dragging forces.

Algorithms for tree drawing have a great advantage since a tree is planar and acyclic. The drawing process can be done iteratively, from the root of the tree to the leaves, assuring that there is no edge crossings. Some of the most relevant techniques for tree drawing based on the node-link diagram are the circular layout (MELANÇON; HERMAN, 1998) and radial layout (REINGOLD; TILFORD, 1981; EADES, 1991). Figure 2 shows examples of radial and circular layouts for a tree.

The algorithms mentioned above are frequently applied in graph and tree drawing. However, those approaches do not construct a reliable layout for several datasets. Furthermore, those models employ only the spatially information, discarding any additional information that can be used in the drawing, such as data attributes. This strategy can generate uncertainty and misinterpretation about the dataset. Thus, there are aesthetic factors, in terms of perception and cognition, that contribute to improve the readability of a graph visualization (BENNETT et al., 2007). Those factors show that only organizing elements through space do not produce a suitable graph or tree layout for many datasets.

Figure 2 – Examples of tree visualization algorithms.



(a) Circular layout (b) Radial layout

Source: (a) Melançon and Herman (1998) (b) Burch et al. (2011) © 2011 IEEE

A graph layout can be transformed in order to improve the extraction of knowledge from the data. Herman, Melançon and Marshall (2000) and Landesberger et al. (2011) presented an extensive collection of techniques that aim to improve the graph representation with different approaches, such as, data filtration, colorization, auxiliary visualization metaphors and user interaction. Those works present many common problems on graph visualization, most of them out of the scope of this research.

In particular, this research addresses the problem of visual clutter in graph visualization. In essence, visual clutter is the result of representing much data in a small area, which reduces the potential usefulness of the visualization. This is a general problem that affects the entire visualization field. Several techniques and applications in a high diversity of domains can suffer from visual clutter (ELLIS; DIX, 2007). In graphs, it is noticed when the number of vertices and edges increases. Visual clutter makes the task of analysing a graph more complex and useless. Some techniques have been proposed to address this problem using strategies like edge filtering, edge clustering, hierarchy representations, space-filling techniques, among others (LANDESBERGER et al., 2011).

One of those approaches that has been attaining success on reducing visual clutter is the group of edge bundling techniques. Edge bundling consists in attract edges towards similar flows. Although it is not necessary, edges are often curved to be easy to follow with the eye. The overlap of multiples edges reduces the clutter and reveals high-level patterns about the data. Edge bundling was already used to reduce the visual clutter in several domains and applications, such as geographic information (CUI et al., 2008; HOLTEN; WIJK, 2009), social networks (MARTINS et al., 2012), scientific papers (ERSOY et al., 2011), multidimensional

projection evaluation (FADEL, 2016), and others (KIENREICH; SEIFERT, 2010). In the next section, we present the state-of-the-art on edge bundling.

2.3 Edge Bundling

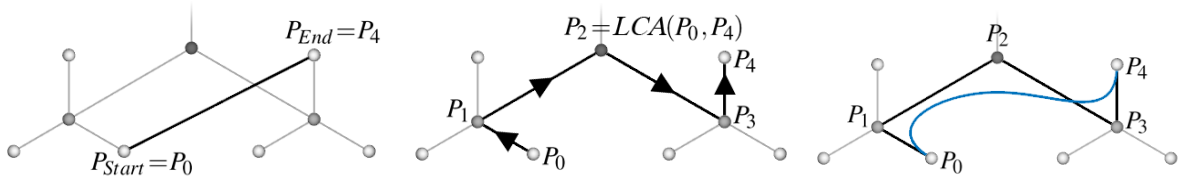
Edge bundling tackles the visual clutter problem by bending edges from the original graph, which generates routes and groups that improves the readability of node-link diagrams (SUN et al., 2013). Curved edges are easy to be followed by the eye (BACH et al., 2016), while the edges overlapping reduces the clutter. In summary, edge bundling is based on a trade-off between clutter and overlap, since it improves edges macro-structures identification, while fewer individual ones are visible (ERSOY, 2013).

The first edge bundling technique is the Hierarchical Edge Bundling (HEB) (HOLTEN, 2006). This technique uses an alternative structure to determine how the edges are bundled. This structure defines a hierarchy amongst all vertices of V, apart from the graph layout and the set of edges E. In detail, this process starts with the drawing of the given structure into the visual space, which can be made by any tree layout algorithm such as the radial, circular, hiperbolic or treemaps (JOHNSON; SHNEIDERMAN, 1991; BRULS; HUIZING; WIJK, 2000). Then, each original edge is drawn as a curve that follows the path between the source and the target vertex in the hierarchy.

In order to transform an edge into a curve, HEB uses the intermediate vertices of the hierarchy, which are not part of the original graph layout. Holten (2006) drawn the edges as cubic B-splines (basis splines) curves, in which the intermediate vertices are set as control points. Figure 3 shows how this process is executed for a given edge. To draw the edge between the original vertices P_{Start} and P_{End} (Figure 3a), the first step consists of finding the least common ancestor $LCA(P_{Start}; P_{End})$ to define a path between these vertices (Figure 3b). Then, the curve is drawn using the intermediate vertices $(P_1; P_2; P_3)$ as control points (Figure 3c).

The cubic B-Spline is a well-known way to draw a curve from a set of points. In summary, a spline is a piecewise function that creates an approximated curve from a sequence of

Figure 3 – The Hierarchical Edge Bundling proposal to bend edges. To draw the edge that connects P_{Start} and P_{End} (a), first the path between the source and the target determine the intermediate points $(P_1; P_2; P_3)$ (b). Then, a curved B-Spline is drawn using $P_1; P_2; P_3$ as control points (c).



(a) Step 1 (b) Step 2 (c) Step 3

Source: Holten (2006) © 2006 IEEE.

coordinates. The curves obtained from a spline can be interpolated splines, when they passes through all the control points, or approximation splines when they do not pass over the control points (HEARN; BAKER, 1986).

B-Splines are approximation splines, suitable for bundling purposes, because the resulting curve is smoother and we can draw several curves that are close but do not exactly overlap each other. In contrast, the interpolated ones are rougher for bundling purposes and would lead us to just draw a curved representation of the hierarchy, once all edges are drawn exactly over the control points. In order to fit approximation curves in the graph layout, we need to perform a slightly modification in the points sets, duplicating the first and the last point. Hence, we assure that all edges will start and finish in the exact position of the source and target vertices.

Holten (2006) conducted an extensive study over B-Splines, Beta-splines and Bézier curves and concluded that there is no significant difference among them. Therefore, the use of a B-Spline is not properly a bundling requirement. Other bundling techniques employ different curves or approaches that do not use any curve construction model. Despite that, Holten (2006) presents a transformation over the spline curve regulated by a tension parameter β. This parameter determines how rigid the edges are drawn, when β tends to 0, the edges are drawn without any distortion (straight), and when β approximates to 1, the edges become rigid curves. Given a curve $S(t)$ over $t \in [0;1]$, starting in P_0 and ending in P_N, this transformation is defined by the following equation:

$$S'(t) = S(t)\beta + (1-\beta)(P_0 + t(P_N - P_0)) \tag{2.1}$$

For a given set of points $P = P_0; P_1; \ldots; P_N$, this transformation can be applied directly over each control point P_i, by the equation

$$P_i' = P_i\beta + (1-\beta)(P_0 + \frac{i}{N}(P_N - P_0)) \tag{2.2}$$

Resulting in the transformed set $S' = \{P_0'; \ldots; P_N'\}$ used to drawn the edges. Figure 4 shows how the value of β modifies the edge bundling visualization.

Furthermore, an edge bundling layout can also be enhanced by color and opacity. A potential usage of color in the edge drawing is to identify edge directions on directed graphs. Edges directions are often represented as arrows. However, it only works for small graphs. Figure 5 shows the visualization of function calls in a source code package using a red-to-green interpolation from the source to the target of each edge. Although this figure has a good representation for directions, the result depends on the dataset features and is not useful when the graph has a more uniform distribution amongst its edges.

Finally, Holten (2006) also evaluated how opacity can be used to improve the bundling visualization. Opacity is commonly used in visualization when there overlapping amongst

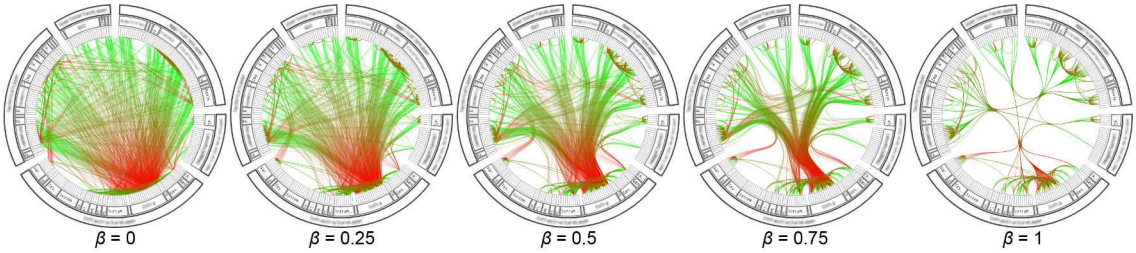
Figure 4 – Different edge bundling layouts showing how the value of β affects the visualization.

(a) β = 0.0 (b) β = 0.5 (c) β = 1.0

Source: Adapted from <http://mbostock.github.io/d3/talk/20111116/bundle.html>

elements, so that it would show elements that were hidden before. Moreover, specifically for the edge bundling layout, combine transparent edges may increase the detection of how many edges are bundled together on each route. Usually, this is a parameter that can be modified by the user. However, Holten (2006) observed that long distance edges overshadowed short ones, so he defines the opacity inversely proportional to the edge length.

Figure 5 – Different examples of a combined usage of gradient color scheme to represent edges directions and values of β.



Source: Holten (2006) © 2006 IEEE.

HEB proved to be a good approach to handle visual clutter in graph visualization that can be applied in real world scenarios. Specifically, it was applied in software visualization (HOLTEN; CORNELISSEN; WIJK, 2007), and in many others domains (TAYLOR et al., 2009; MANSMANN et al., 2009; GOU et al., 2011). Its achievements created a new branch of graph visualization, which became an interesting field for many researchers. However, HEB requires a hierarchical structure to conduct the bundling process, which might not be properly found in many datasets and there is no discussion about methods that create this hierarchy. In addition, HEB also does not work for geometric based graphs, since it demands a point placement that considers the hierarchy, not the set of original vertices.

In the next sections, we outline different edge bundling approaches that were presented after HEB. In order to have a better organization, these techniques were grouped by their most

remarkable aspect: geometry-based, force-based, image-based, or attribute-based approaches.

2.3.1 Geometry-based Edge Bundling

In this section, we present edge bundling techniques that use as primary concept any kind of geometric information from the original graph layout to bundle edges. Therefore, instead of using an additional hierarchy or other information, these techniques require an original graph layout with vertices already placed in the visual space. These techniques are the Geometry-based Edge Clustering (GBEC) (CUI et al., 2008), the Winding Roads (WR) (LAMBERT; BOURQUI; AUBER, 2010), and the Multilevel Agglomerative Edge Bundling (MINGLE) (GANSNER et al., 2011).

As mentioned before, one of HEB major problems is that it needs a hierarchy and the vertices placement is based on the hierarchy layout. Hence, this makes HEB unable to handle graphs that vertices coordinates must be retained. The GBEC is the first technique that successfully achieved such goal. This technique is an extension of Qu, Zhou and Wu (2007) that builds a mesh based on the original graph and it uses this mesh to bend the edges. The resulting layout quality is defined by the quality of the applied mesh, so the major problem with this approach is how to determine a good mesh.

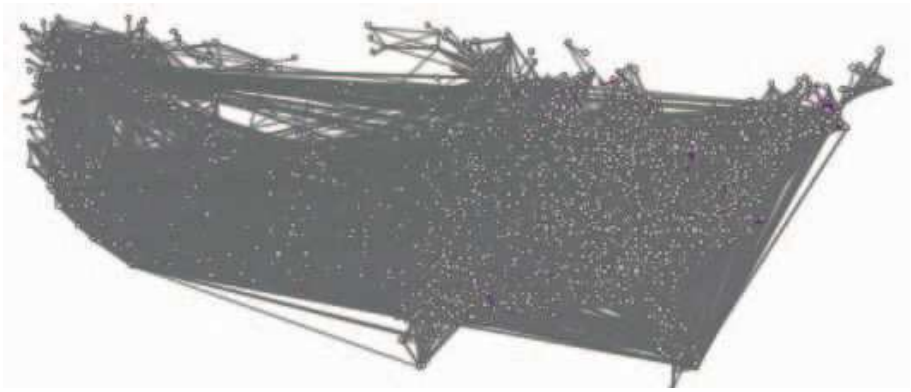
The mesh construction method is not trivial because it needs to reflect the underlying graph structure in order to create a road-map of edges. The strategy presented by Qu, Zhou and Wu (2007) consists of building a mesh using a Delaunay triangulation of nodes. However, Cui et al. (2008) showed that this approach can not handle large graphs and it often produces unpleasant layouts. A straightforward way to create this mesh is a manual process, managed by the user after some observation of the graph structure, in which edge densities and directions are analyzed. However, this can be a complex and time-consuming task, in special for large graphs. In addition, it is not guaranteed that the user will provide a useful mesh.

In order to provide an automatic mesh generation, Cui et al. (2008) performed an analysis of edge patterns through the original graph by a discretization of the visual space. This process computes the original graph bounding-box and divides the resulting area into cells. After that, each cell is analyzed in terms of its density of elements and edges directions. Next, small regions with similar direction are merged to construct larger regions, thus forming clusters. At the end, the mesh can be constructed applying a constrained Delaunay triangulation on segments that perpendicularly cross each cluster. Using this process, the variation of direction when merging small cells into larger ones can produce different levels of details.

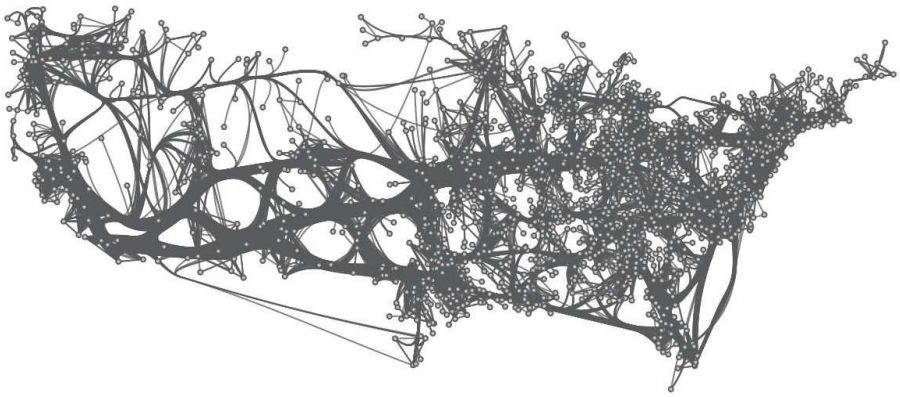
Finally, the edge bundling layout is constructed by setting control points and drawing segments or curves through them. These points are determined by the intersections among the mesh and the original edges, with close points being merged. Figure 6 shows the GBEC result for the dataset United States airlines, a graph with 1,790 vertices and 9,798 edges representing

flights between different US cities. The set of vertices is placed into the visualization according to their corresponding geographic location. This was not possible with HEB.

Figure 6 – A GBEC layout for the dataset United States airlines, with 1,790 vertices ans 9,798 edges representing flights between different US cities. It is possible to notice that edges are bundled into "road-maps".



(a) Original graph



(b) Bundled graph

Source: Cui et al. (2008) © 2008 IEEE.

In a similar approach, the technique Winding Roads (WR) (LAMBERT; BOURQUI; AUBER, 2010) also discretizes the visual space to determine edges routes. This technique has a process similar to a traffic system, in which different segments are larger or shorter depending on the number of vehicles they support. Thus, this technique creates routes and then transform each original edge to follow the most suitable route for its direction.

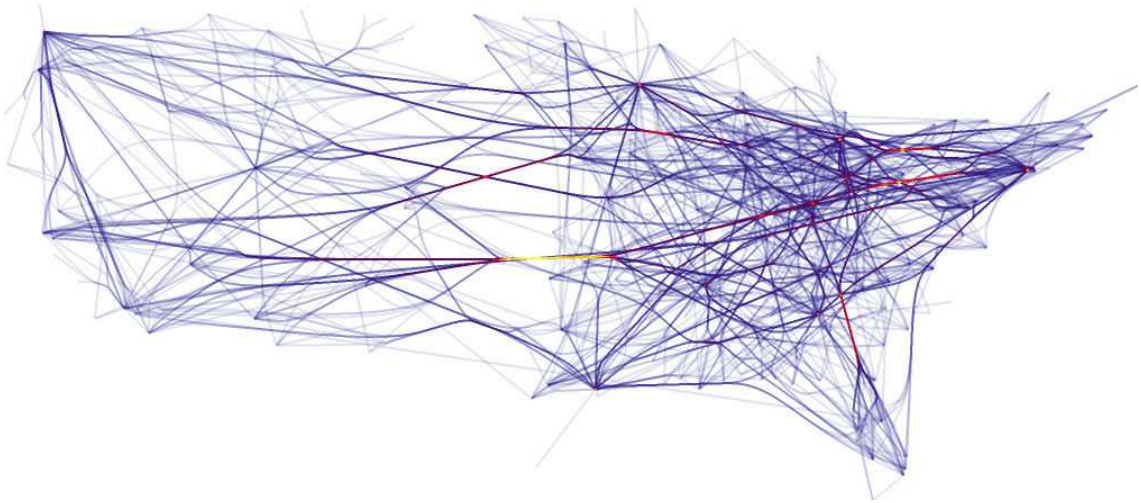
In detail, WR first divides the space into a grid of cells considering each vertex position. This process uses a hybrid approach based on QuadTree decomposition (FINKEL; BENTLEY, 1974) and the Voronoi diagram construction (VORONOÏ, 1908), to create a "city map" over the original graph. Next, using the Dijkstra algorithm, each original edge is transformed into the shortest path between its source and target in the grid. This process assures that the final graph will not have edge crossings. Finally, the edges are smoothened using Bezier Curves. The control points are determined by the crossings between the original edge and the grid. WR produces better edge bundling layouts (in terms of clutter reduction) and it runs faster than the GBEC.

The last technique in this group is the Multilevel Agglomerative Edge Bundling (MINGLE) (GANSNER et al., 2011). Different from previous techniques, that share the same general strategy, MINGLE employs an approach that aims to minimize the density of elements in the visual space. To achieve it, the technique performs an iterative edge clustering computation and calculates the usage of ink to draw the graph. This process is an extension of an edge clustering method to improve circular layouts (GANSNER; KOREN, 2007). It can be computed faster than previous algorithms and it can be applied in larger graphs.

To create an edge bundling layout, MINGLE uses an auxiliary structure, which is called graph of edges proximity, computed from the compatibility among edges from the original graph layout. Each edge is formed by a point in a 4-dimensional space, consisting of the coordinates of the edge source and target. Thus, original graph is processed to determine connections among each edge with its k nearest edges in the graph of edges proximity, The parameter k can be used to determine how many edges will be bundled together. Once the proximity graph is constructed, it will guide the bundling decisions.

The bundling process consists of merging edges in order to minimize the amount of ink used to drawn the entire graph. This computation is performed considering the proximity graph. For each adjacency (i.e., for each pair of neighboring edges from the original graph), the amount of ink to draw the graph is calculated considering that the pair was bundled. Then, the pair with maximum ink saving is chosen to be bundled. When edges are bundled, they are also grouped in the proximity graph. The next step can join not only pairs of edges, but also groups that represent bundles processed before. The rendering process consists in drawing each bundle of edges as a unique line and connects this line with the original vertices at its endings. Finally, the lines are transformed into splines to smooth out the drawing. Figure 7 shows the edge bundling layout performed by MINGLE for the airlines graph.

Figure 7 – A MINGLE layout for the dataset United States airlines.



Source: Gansner et al. (2011) © 2011 IEEE.

Among the three techniques presented in this section, MINGLE has the lowest computational cost, and it is the first technique to perform edge bundling in graphs with millions of edges. However, there is no further discussion on graph readability (for example, if the bundling can really increase the readability of millions of edges). The ink saving measure can be used to make the MINGLE bundling decisions, but not to compare different bundling techniques, since it is not possible to conclude that lesser the ink usage is, better is the graph layout.

Furthermore, those techniques only use the geometric information, by the grid computation or the proximity graph from the original edges placement, to execute the bundling task. It means that there is nothing else about the data are being considered in order to create edge bundles. In the next section, we present other group of techniques, that employs force-based methods to bundle edges.

2.3.2 Force-based Edge Bundling

Force-based approaches cover a large domain of techniques in information visualization, such as multidimensional projection (TEJADA; MINGHIM; NONATO, 2003) and text visualization (ALSAKRAN et al., 2012). These approaches simulate a force system, in which each element in the visual space is a point with attractive and repulsive forces to other elements, the elements move over the space until the equilibrium state is reached. In edge bundling, the force-based approach is an alternative to more complex mesh generation layouts. This group consists of two main techniques, the Force-Directed Edge Bundling (FDEB) (HOLTEN; WIJK, 2009) and the Divided Edge Bundling (DEB) (SELASSIE; HELLER; HEER, 2011). FDEB segments each edge into a set of points to build a spring–mass system and executes an iterative process to bend the edges. This process groups similar edges until the system reaches a stable state. DEB improves the FDEB layout by separating edges with different directions, improving the readability of directed graphs.

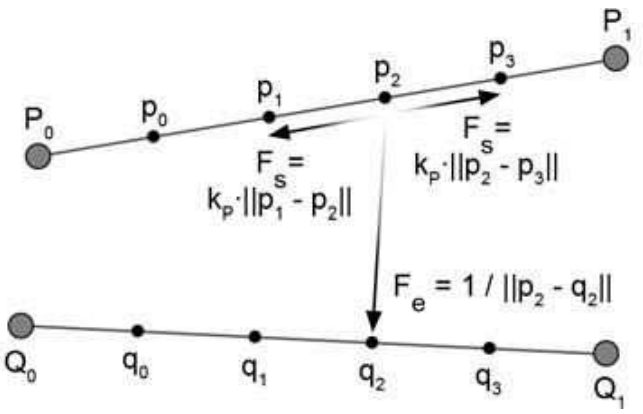
The method employed by FDEB consists in a spring–mass system, which combines a spring compressing force and an electrostatic force. First, each edge is segmented into k intermediate points $P_{i;j}$, where i is the edge index and j one point ($j \in [1;k]$) along its length. Next, the forces system is built with each point $P_{i;j}$ being affected by a spring force in the direction of its neighboring points ($P_{i;j-1}$ and $P_{i;j+1}$), and an electrostatic force for each other point $P_{s;j}$ with $s \neq i$. Figure 8 shows how this process works, considering the segmentation of two edges and the forces applied to one intermediate point. In each step of the iterative process, all points are moved towards the resulting force, which is computed by the following equation:

$$F_{P_{i;j}} = K_i.(\|P_{i;j-1} - P_{i;j}\| + \|P_{i;j} - P_{i;j+1}\|) + \sum_{s \neq i} \frac{1}{\|P_{i;j} - P_{s;j}\|} \tag{2.3}$$

with K_i being the spring constant for the edge i and $P_{s;j}$ the intermediate point j of the edge s. The edge bundled graph is obtained when this system reaches its state of equilibrium, by drawing

splines using the intermediate points as control points. The authors limited the interaction among control points to only those with the same index to reduce the computational cost of the simulation.

Figure 8 – Forces involved to determine the control points in the Force-Directed Edge Bundling. In detail, the figure highlights the forces that determine the position of the intermediate point p_2. These forces represent a spring force in the direction of p_1 and p_3 and an electrostatic force in the direction of q_2.



Source: Holten and Wijk (2009) © 2009 IEEE.

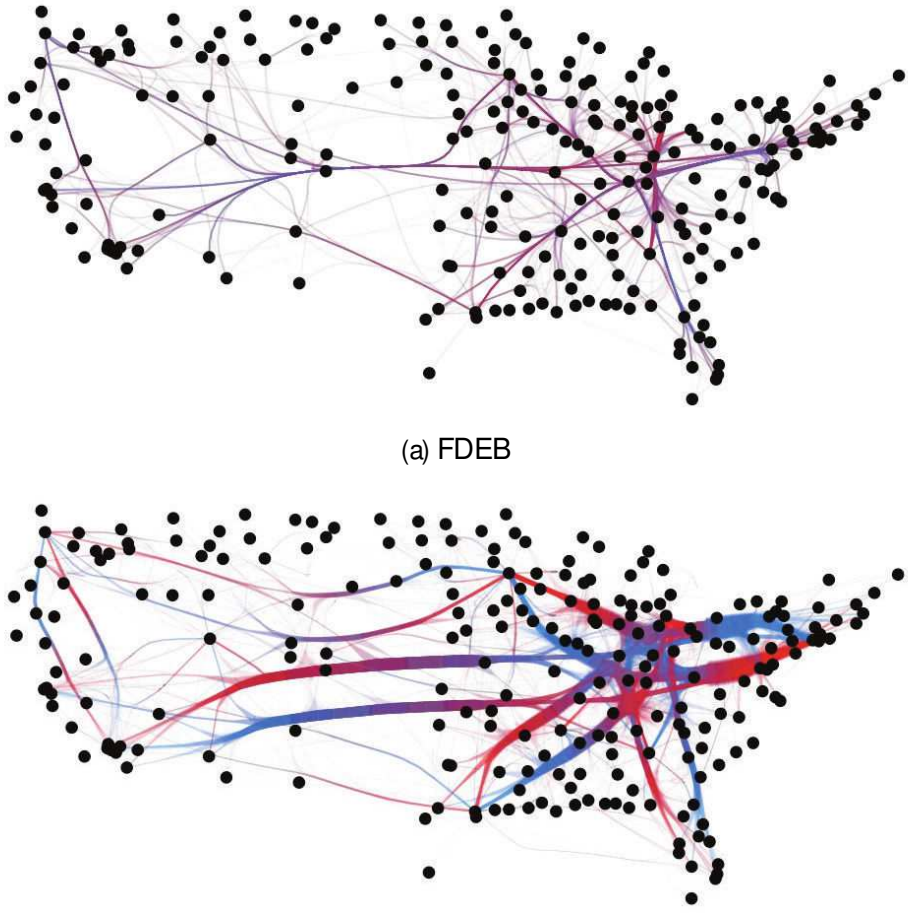
However, the resulting graph is often unreadable due to the strength of the bundles. Varying the spring constant K_i is not useful to solve this problem since it relaxes bundles too much. So, Holten and Wijk (2009) employed an edge-compatibility measure $C_{i;s}$ into the force system, which controls the level of interaction between each pair of edges. This measure can consider different factors, such as the angle, size, position and visibility of the edges. Therefore, the final formulation is given by the equation:

$$F_{P_{i;j}} = K_i.(\|P_{i;j-1} - P_{i;j}\| + \|P_{i;j} - P_{i;j+1}\|) + \sum_{s \neq i} \frac{C_{i;s}}{\|P_{i;j} - P_{s;j}\|} \quad (2.4)$$

Another benefit of the compatibility measure is that it is a flexible metric. Basically, it is possible to add any feature that measures the relationship between edges, including related information about the edges, such as attributes or weights. Divided Edge Bundling (DEB) is an example of the manipulation of this compatibility measure. Selassie, Heller and Heer (2011) modified the forces in the physical simulation in order to highlight edges directions and weights. Figure 9 shows a comparison between FDEB and DEB for the same dataset. It is possible to see how DEB better represents the bundling directions and weights.

Both force-based approaches presented an important contribution to edge bundling layouts by creating clean and meaningful edge routes. However, force-based techniques have a high computational cost. FDEB is $O(E^2C)$ for a graph with E edges and C control points per edge for every iteration. DEB has pre-processing tasks that reduce the complexity to $O(E^2)$ per iteration. Also, those techniques do not describe deeply or evaluate the usage of underlying data

Figure 9 – A comparison between the FDEB and DEB layouts for the United States airlines. This comparison shows how DEB better represents the edges directions.



(a) FDEB

(b) DEB

Source: Selassie, Heller and Heer (2011) © 2011 IEEE.

into the compatibility measure. This is only addressed in other publications (NGUYEN; HONG; EADES, 2012; SAGA; YAMASHITA, 2015; YAMASHITA; SAGA, 2015).

In the next section, we discuss the group of image-based edge bundling techniques. Those techniques reduce the computational cost and running times by using image processing methods.

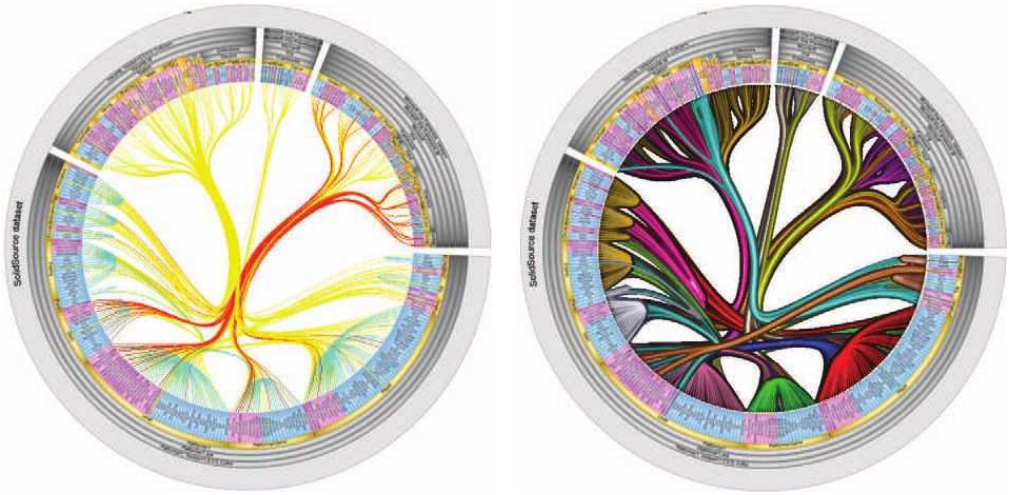
2.3.3 Image-based Edge Bundling

Using a different strategy, a group of techniques relies on image processing approaches to produce and improve edge bundling layouts. There are four techniques in this group, the pioneer is the Image Based Edge Bundles (IBEB) (TELEA; ERSOY, 2010), that processes an existing image of a bundled graph to improve the visual representation and provide a better perception on groups separation. The second technique is the Skeleton-Based Edge Bundling (SBEB) (ERSOY et al., 2011), that employs a set of image processing algorithms to create paths that guide the

edges bending. The Kernel Density Estimation Edge Bundling (KDEEB) (HURTER; ERSOY; TELEA, 2012) draws bundled edges using the kernel density estimation algorithm, rendering a bundling layout much faster than prior techniques. Finally, the bundling process is further speed up by the technique CUDA-based Universal Bundling (CUBu) (ZWAN; CODREANU; TELEA, 2016), which is an extension of KDEEB that takes advantage of Graphics Processing Unit (GPU) parallelization.

The IBEB is not a properly novel edge bundling technique because it does not present a bundling construction algorithm. Instead, this method presents an image-based transformation to enhance the edge bundling visualization, emphasizing edge clusters and patterns. This technique receives as input an edge bundling layout built with HEB or other techniques. The original set of edges is divided into different clusters by their distances. Then, several image processing methods are applied, such as to determine contours and shadows of each cluster. The bundles also receive distinct colors. Distinct groups can be moved to reduce the overlap among them, which was not possible in prior techniques. Figure 10 shows the enhancement obtained by employing the IBEB in a original HEB layout.

Figure 10 – A comparison between HEB layout and its enhanced version using the IBEB. The IBEB highlight edge bundles and remove the occlusion among them.



(a) HEB (b) IBEB

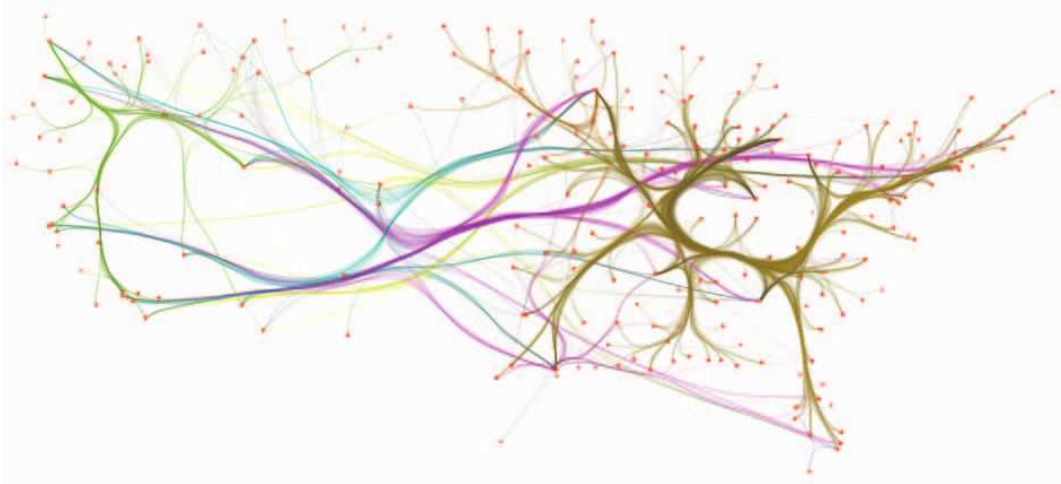
Source: Telea and Ersoy (2010) © 2010 The Author(s) Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

Extending the IBEB, the Skeleton-Based Edge Bundling (SBEB) (ERSOY et al., 2011) technique proposes an entire image based framework to construct edge bundling layouts. Instead of receiving a pre-processed graph, the technique receives as input the original graph layout, where vertices are already placed in the visual space. This technique uses similar IBEB iterative image manipulation methods to create the bundling visualization. During the bundling process, the technique takes advantage of GPU processing to create bundling layouts faster than previous methods.

In detail, SBEB uses a skeleton computed from the original graph to guide the bundling. This process resembles the HEB, but it does not require any additional information apart from the original graph. The skeleton is inherent to the graph topology and computed in the first stage of the framework. The whole process of SBEB is complex and it takes several steps, which are beyond the scope of this section. Briefly, in order to create the skeleton, the edges are first divided into clusters according to their geometry similarity, such as in IBEB. Next, the technique obtains the shape of each cluster and a skeleton is computed from each shape. Finally, edges are attracted towards their respective cluster skeleton.

Although the technique produces a suitable layout after one execution, the process can be repeated iteratively to improve the visualization. Moreover, other minor enhancements are applied, such as relaxation and smoothing of the edges. A Compute Unified Device Architecture (CUDA) based implementation of SBEB performs the bundling of small graphs in few milliseconds. Figure 11 shows the United States airlines graph produced by SBEB, colors were used to identify each edge cluster.

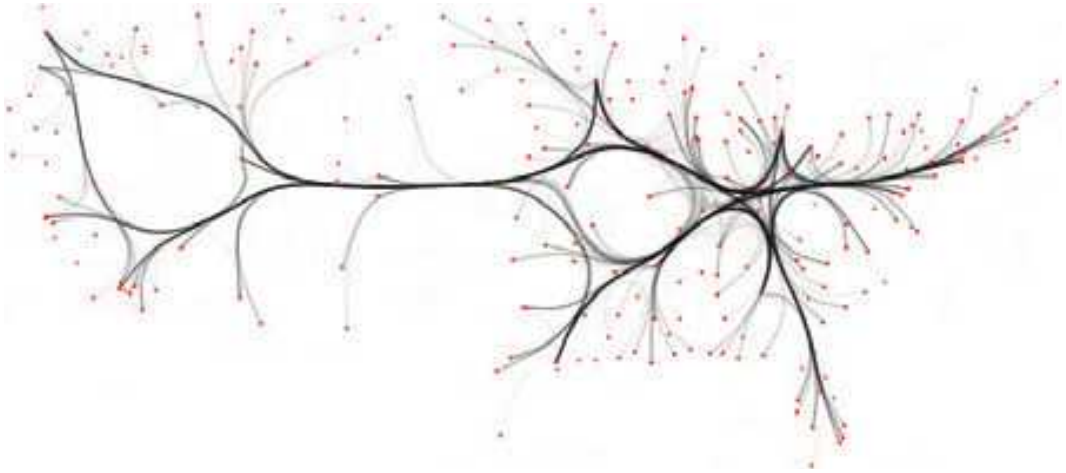
Figure 11 – The edge bundling layout produced by the SBEB for the dataset United States airlines.



Source: Ersoy et al. (2011) © 2011 IEEE.

The third image-based technique is the Kernel Density Estimation Edge Bundling (KDEEB) (HURTER; ERSOY; TELEA, 2012). This technique uses an approach based on the Kernel Density Estimation algorithm. Basically, it generates the density map of a given graph layout. Then, it transports the edges in the gradient of this map in an iterative process, reducing the kernel size at each step. Similar to SBEB, the input graph can be the original or bundled graph. This approach allows the user to modify the density map, for example, by adding obstacles that must be avoided in the bundling construction. Finally, the technique also improves the shading algorithm from former image-based techniques, used to enhance the visualization. KDEEB produces similar results to the previous techniques in terms of clutter reduction and bundling quality, while it runs in a fraction of time of prior algorithms.

Figure 12 – A KDEEB layout for the dataset United States airlines.



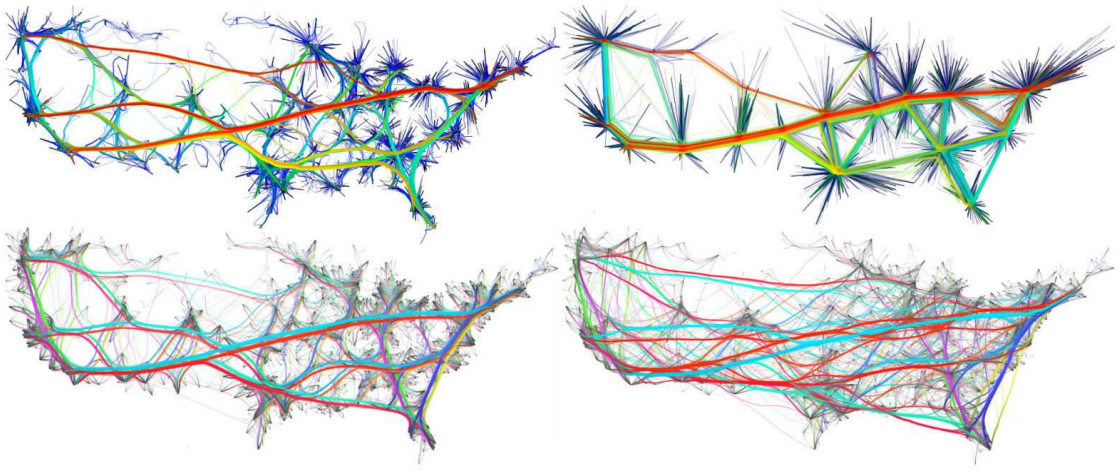
Source: Hurter, Ersoy and Telea (2012) © 2012 The Author(s) Computer Graphics Forum © 2012 The Eurographics Association and Blackwell Publishing Ltd.

The last image based technique is also the most recent edge bundling publication, called CUDA-based Universal Bundling (ZWAN; CODREANU; TELEA, 2016). As its name says, this technique consists in a completely parallelized GPU-based approach to construct edge bundling layouts. This approach runs faster than KDEEB, which was the quickest bundling implementation so far. Furthermore, CUBu claims to improve scalability, edges direction representation, the level of details and the generality for edge bundling.

In detail, CUBu algorithm receives as input an original graph layout, like other image-based techniques. CUBu's bundling approach originated from the KDEEB algorithm without major changes. It also obtains a density map from the kernel density estimation algorithm. However, CUBu improves each step to a more parallelized strategy. For instance, the density map takes between 40% and 60% of the execution time of KDEEB and it can not be completely executed in parallel due to concurrent image-writes. Another difference is how frequent the smoothing process is applied, one for each 3 or 4 iterations, instead of one for each iteration in KDEEB. Despite the similar bundling algorithm, CUBu presents improvements in the bundling shape control, the representation of edges directions and a set of original options that can be used to change the final layout. Figure 13 shows an edge bundling layout generated with CUBu.

KDEEB and, recently, CUBu are considered the state-of-the-art on edge bundling techniques. Comparing these techniques, KDEEB processes a graph with 738,491 vertices and 899,791 edges in 8 seconds, while CUBu takes 0.152 seconds for the same input, according to the results reported in Zwan, Codreanu and Telea (2016). Both techniques are also used for real-time dynamic bundling. These techniques have been proposed only to address the poor computational performance of prior techniques, and there is no discussion about the faithfulness of the bundling representation considering the input data. This is a common practice on most edge bundling publications, although some new techniques discuss how the underlying data can improve edge bundling layouts. Those techniques are presented in the next section.

Figure 13 – A CUBU layout for the dataset United States airlines using different configuration parameters.



Source: Zwan, Codreanu and Telea (2016) © 2016 IEEE.

2.3.4 Attribute-based Edge Bundling

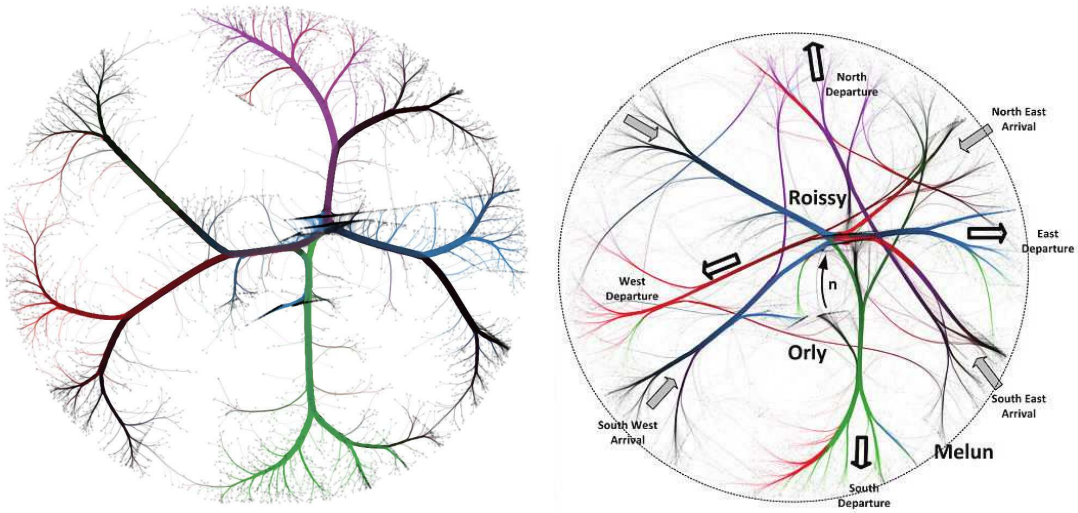
Although previous methods produced good results regarding running times, which allowed the processing of thousands of edges in real-time, the layout meaningfulness is neglected when information on edges or vertices (data attributes) is ignored. This information is not used to guide the bundling process, so there is no connection between the visual representation and the underlying data.

a few techniques define strategies that consider some information. Hierarchical Edge Bundling consider underlying data by means of a hierarchy. However, it does not discuss any alternative way to construct this structure from the data. It presents results only for software based graphs, which used the software structure, a natural hierarchy, to generate the bundling layout. Another common approach is, as mentioned before, to modify the Force-Directed Edge Bundling compatibility measure. FDEB has been extended to use semantic properties inherent from edges (KIENREICH; SEIFERT, 2010), edge type or attributes (YAMASHITA; SAGA, 2015), and compound compatibility measures (NGUYEN; HONG; EADES, 2012) on the force model calculation.

Recently, Attribute-Driven Edge Bundling (ADEB) (PEYSAKHOVICH; HURTER; TELEA, 2015) extends the image-based KDEEB by using edge attributes to set the bundling flow map. ADEB uses the same framework from KDEEB, but creates different density maps for each edges attributes. This process basically adds the compatibility modifier transformation to KDEEB and it can only be processed for numerical edges attributes. The results presented show its usefulness to separate bundles according to edges directions, which can not be achieved with KDEEB. Figure 14 shows a comparison between KDEEB and ADEB.

In all presented cases, only the edge information is considered to improve edge bundling techniques. The information contained on vertices is still ignored. Moreover, one of the most

Figure 14 – A comparison between the KDEEB and ADEB layouts of the Paris Air Traffic dataset. This result shows that ADEB uses edges weights to create better bundles. Using this information, the ADEB layout can separate flights between departures and arrivals for each direction.



(a) KDEEB (b) ADEB

Source: Peysakhovich, Hurter and Telea (2015) © 2015 IEEE.

important open problems in edge bundling is to determine how faithful is an edge bundling representation (NGUYEN; EADES; HONG, 2013a; LHUILLIER; HURTER, 2015).

2.4 Final Remarks

In this chapter, we presented the techniques that turned edge bundling into one of the most successful graph drawing transformations to reduce visual clutter. Edge bundling techniques have been used in the last 10 years in many fields, proving that they deliver a great way to enhance graph visualization and improve visual analytics tasks. Also, edge bundling has been consistently improved and extended by new techniques during all this period.

By taking advantage of parallel and graphics processing, recent edge bundling techniques reached a superb performance for large graphs, allowing them to be used with graphs containing thousands, or even millions, of vertices and edges, and to compute real-time changes, as in dynamic graphs.

Despite that, there are still concerns about edge bundling visualization. In general, the goal of the existing bundling techniques is to simplify the visual representation of a graph, emphasizing the main topological patterns presented in the entire dataset. The edge bundling techniques presented in this chapter did not have a suitable validation process to measure how faithful was a given edge bundling layout, considering the given data. As presented, most techniques create roadmaps, but do not consider the real meaning of drawing a set of edges in the same route, or what the user can interpret from that.

For example, this chapter presented edge bundling layouts from the dataset United States airlines constructed by six different techniques (GBEB, MINGLE, DEB, SBEB, KDEEB and CUBu). In all results, the clutter reduction is easily observed, but there is no consensus on which technique provided the best result. Interpret a bundling layout is often difficult, because the user has no information that explains what a bundle represents in the original data.

This master's thesis aims to fill this gap, by presenting a suitable bundling framework that handles the information surrounding vertices and edges in order to determine how the edges are bundled. Different from geometry-based techniques, we do not use the graph's spatially data to create bundles, because it can lead the visualization to show misleading information. Instead, we attract the edges over a similarity-based structure, called backbone. This is also different from the force-based approach and its alternatives that may consider data not only spatially data, but merge all information in the same system.

In addition, prior techniques, such as GBEB, MINGLE and iterative image-based techniques, support multiple levels of details. However, those methods just vary how generic the bundles are. For example, MINGLE determines it by the maximum amount of edges that can be bundled together. On the contrary, this research explores a multiscale visualization based on vertices filtration, which can separate global and local patterns. Hence, improving the visualization of huge graphs. Therefore, we discuss the application of our technique in huge graphs by its ability to present a readable graph, not by its computational cost.

CHAPTER

3

SIMILARITY EDGE BUNDLING FRAMEWORK

3.1 Initial Remarks

In the previous chapter, we reviewed the existing techniques, discussing their most important points and drawbacks. Currently, as discussed, most techniques focus on improving the computational scalability, reducing the computational costs. However, not much has been made to prove if the generated bundles can faithfully display the information of the original data.

In this chapter, we present our framework for similarity-driven edge bundling. The idea of adding similarity to the process to improve its meaningfulness is based on the Gestalt principle of proximity (STERNBERG, 2008). Gestalt is a well-known set of psychology laws and principles that aims to understand the human ability to perceptually acquire and maintain information, being a fundamental concept in the field of information visualization. One of those principles states that the human mind, when processing a visual representation, usually associates the similarity among elements by their proximity (KOFFKA, 2013). Although this principle is important for bundling purposes, it has been ignored by current techniques.

Moreover, it has been a consensus that edge bundling techniques are capable of reduce clutter while representing generic information, being often applied to present a preliminary insight into the data (HOLTEN, 2006). However, it is also a consensus that edge bundling techniques presents problems when handling large datasets, since the cluttering, although reduced, will persist when graphs with many nodes and/or edges are bundled.

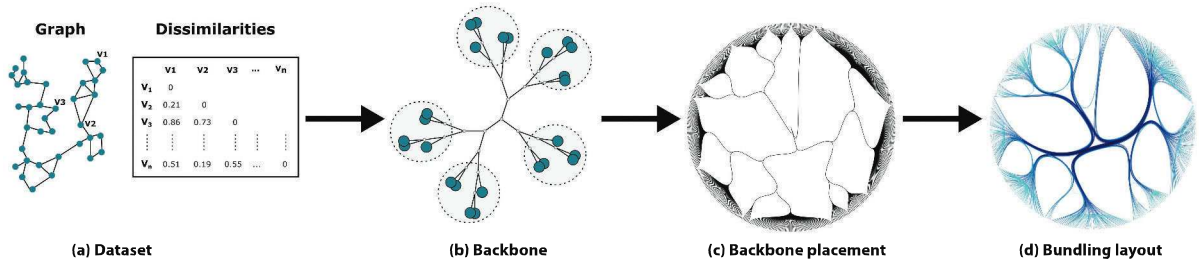
From those assumptions, we consider to improve edge bundling following two different principles. First, taking the similarity among elements into account at the moment we pack the edges into bundles, and second, creating strategies to display the information over different levels of detail. This research supports the following hypothesis:

> "Edge bundling can better represent the data when there is an inherent connection between the proximity among the elements in the information space and the proximity of edges in the visual space. Moreover, providing a multiscale representation improves the visual and complexity scalability of bundling layouts, making them able to represent more information and to handle bigger datasets."

In this context, our goal is to create multiscale edge bundling layouts using the distance information contained on the dataset. In other words, given a graph $G = (V;E)$ composed of a finite set V of vertices and a finite set E of edges, with a vertex $v_i \in V$ representing a data object $d_i \in D$, and an edge $e_{ij} = \{v_i;v_j\} \in E$ representing some relationship between different vertices $v_i \in V$ and $v_j \in V$. Moreover, suppose that $\delta(d_i;d_j)$ is a function that measures the dissimilarity between two data objects d_i and d_j. Our goal is, when drawing this graph, to bundle edges creating groups that obey the dissimilarity relationships amongst the vertices, i.e., the data objects they represent. To reach this goal, we developed the Similarity Bundling Framework.

Our framework defines some steps in order to compose similarity driven bundles. In this process, we employ a structure called backbone to create a "road-map" of bundles and, finally, draw edges following these roads. In detail, our methodology splits the edge bundling construction into three different steps. The first step consists of the backbone construction, in which we build a structure that will guide the bundling. In the second step, we place the original vertices and the backbone into the visual space. Finally, the third step involves bending the edges from the original graph into bundles. Figure 15 presents an overview of our methodology.

Figure 15 – Similarity Bundling Framework overview. (a) represents the input graph and its matrix of dissimilarities, (b) shows the backbone created from the dissimilarities, (c) shows the backbone placement using the radial layout, (d) shows the bundling layout with edges bent through the backbone.



Source: Elaborated by the author.

In the next sections, we detail each step, starting with the backbone construction.

3.2 Backbone Construction

The backbone is a structure composed by an additional set of elements, the intermediate vertices V', linking all vertices V of the graph G. From now on, vertices V are called data vertices,

i.e., vertices that represent data objects. Every pair of vertices $v_i \in V$ and $v_j \in V$ is linked through a sequence of vertices (path) $p_{ij} = \{v_i; v'_1; v'_2; \ldots; v_j\}$ passing through different intermediate vertices $\{v'_1; v'_2; \ldots\} \in V'$. The primary function of the backbone is to serve as a guide to bend the edges, resulting on the bundles. If an edge e_{ij} connects two vertices v_i and v_j, e_{ij} will be curved towards the path p_{ij} that links v_i and v_j in the backbone.

Different paths can share intermediate vertices. Therefore, the bundle process results in groups of curved edges, potentially reducing the visual clutter. In our approach, the backbone is a key piece and has to obey the following design principles to fulfill its role:

1. The backbone should define paths between all pairs of data vertices, precisely connecting the vertices according to the similarity among them (data objects);

2. Any two data vertices should be linked by a unique path;

3. A path between two data vertices should not pass by any other data vertex.

The first and the second principles are straightforward. Because the backbone is used to attract the edges during the bending, it is necessary to have routes connecting all pairs of data vertices, considering that the original graph could have edges between any pair of data vertices. In addition, the path connecting two data vertices should be unique in order to avoid ambiguity problems during the bending. The third principle is broader. Suppose that an edge e_{ij} connects the vertices v_i and v_j, but that neither v_i nor v_j are linked to a third vertex v_k. Since the backbone is used to attract e_{ij} towards the path p_{ij}, if $v_k \subset p_{ij}$, the edge will be bent towards v_k. This gives the wrong impression that v_i and/or v_j are connected to v_k, potentially resulting in misleading visual representations.

As a consequence, our backbone is a tree-like structure where the data vertices V of the graph G are the leaves. Among the candidate techniques to construct such tree, the minimum spanning tree (GRAHAM; HELL, 1985) can be discarded since it violates (3). We can also discard hierarchical clustering techniques, such as the Unweighted-pair Group Method with Arithmetic Means (UPGMA) (SOKAL; MICHENER, 1958) (SOKAL; MICHENER, 1958), since they are extremely sensitive to certain distance distributions (LEMEY; SALEMI; VANDAMME, 2009), and prone to produce unbalanced structures, which violates (1).

We developed two backbone constructions that obey all design principles. First, we used the Neighbor-Joining (NJ) (SAITOU; NEI, 1987) algorithm for phylogenetic tree construction, and then we improved this approach to add the multi-level capability defined in our hypothesis. In sequence, we developed a novel algorithm, called Similarity Tree (STree), for the construction of the backbone that addresses all the design principles, which creates a multi-level similarity-based binary tree with a low computational cost. We describe these methods in the next sections.

3.2.1 The Neighbor-Joining Backbone Construction

The Neighbor-Joining (NJ) (SAITOU; NEI, 1987) phylogenetic tree was the first approach we used to build the backbone in our edge bundling method. This tree is also used to place the vertices in the visual space. Thus, the resulting graph has its edges grouped, and the nodes placed considering the similarity relationships existent on the dataset.

The NJ algorithm creates a tree containing vertices that represent all N data elements and $N-2$ intermediate vertices, that is, vertices that are not part of the original dataset, but were created by the algorithm. This method starts with a pair-wise distance matrix D and a star-like tree, in which all data vertices are connected to one virtual vertex. Then, the algorithm finds a pair of vertices (i, j) by the criterion of minimum evolution, that is, the pair with the smallest sum of branch lengths $S_{i,j}$ given by the following equation

$$S_{ij} = \frac{1}{2(n-2)} \sum_{k=i,j}^{n} (D_{ik} + D_{jk}) + \frac{1}{2} D_{ij} + \frac{1}{n-2} \sum_{(k,l=i,j) \wedge (k<l)}^{n} D_{kl} \tag{3.1}$$

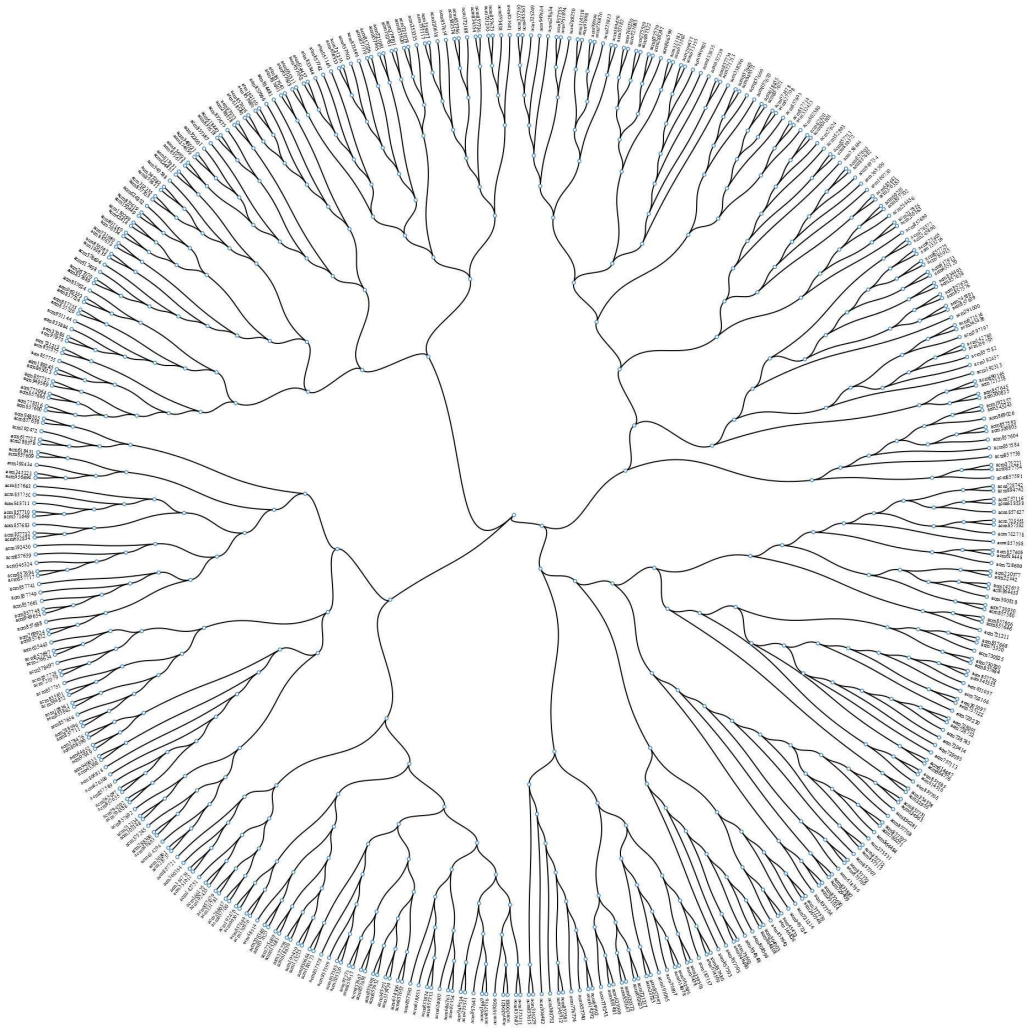
When the pair (i, j) is selected, a new intermediate vertex X is created, with the vertices i and j as its children and connected to the common ancestor of i and j. We then update the distance matrix D by removing the distances that involve the vertices i and j and adding the distances from X to each remaining vertex (k). To calculate these distances, we use the following equation

$$D_{X,k} = \frac{D_{ik} + D_{jk}}{2} \tag{3.2}$$

This process finishes when only two nodes are remaining in D. In the end, the neighbor-joining tree will have N data vertices and $N-2$ intermediate vertices. We used the implementation proposed by Studier, Keppler et al. (1988), in which the NJ is obtained with complexity $O(V^3)$, where V is the number of vertices. Although most usages of the NJ algorithm are related to biological studies, it was applied in information visualization before. Cuadros et al. (2007) created a NJ from collections of documents to perform a visual analysis of text similarity. Despite the requirement of a distance matrix, this matrix can be generated by any data associated with the vertices of the original graph. Figure 16 shows the neighbor-joining tree using the radial layout for the dataset INFOVIS04 (FEKETE; GRINSTEIN; PLAISANT, 2004) (see section 5.2).

The NJ is suitable to construct the backbone because it avoids ambiguities in the bundling visualization. This happens because the algorithm adds intermediate vertices to the tree, and it guarantees that the path between two different data vertices passes only through intermediate vertices, which is crucial to fulfill our design principle (3). Paiva et al. (2011) extended the NJ algorithm in order to promote data vertices in the hierarchy, thus removing intermediate ones. However, this modification would produce a backbone that does not obey this principle.

Figure 16 – The neighbor-joining tree of the dataset INFOVIS04.



Source: Elaborated by the author.

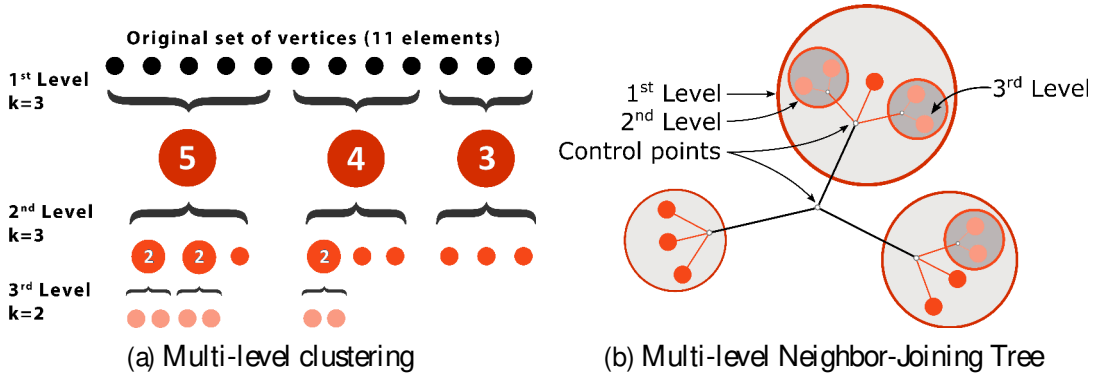
3.2.2 Multi-level Neighbor-Joining Backbone

We transformed the NJ backbone construction into an iterative process that combines the NJ method with a clustering process, such as presented by the Visual Super-Tree (SILVA et al., 2016). In this approach, instead of building the NJ tree from the entire set of vertices, we first split the data vertices into clusters, repeating it in multiple levels, and then construct the tree from the clusters.

Figure 17 shows how the multi-level clustering is applied to construct the bundling backbone. First, starting with the original set of vertices V, we create a set $C^1 = \{c_1^1; c_2^1; \ldots; c_K^1\}$ with K clusters at the first level $l = 1$, which is obtained using the K-means (JAIN; DUBES, 1988) algorithm. Then, we compute the neighbor-joining from the set of centroids $\tilde{C^1} = \{\tilde{c_1^1}; \tilde{c_2^1}; \ldots; \tilde{c_K^1}\}$ of C^1 to start our backbone. Next, the same process is executed recursively. For each level l, we divide each cluster c_i^l into a new set of Q clusters and compute the neighbor-joining of this set. Finally, this tree is attached to the branch where c_i^{l-1} was positioned in the superior level

tree. The process stops when it reaches the desired number of levels. To create the backbone, the user sets the value of K, and the minimum size for a given cluster in the last level. Hence, if a cluster reaches the minimum size before L levels, the process is stopped in this branch and the remaining elements are directly attached to their respective superior tree level.

Figure 17 – Multi-level clustering processing. (a) shows the division of an original set of 11 vertices into three clusters with size 5, 4 and 3 in the first level. Then, the second level divides each cluster in three deeper ones. On the last level, only clusters with more than one element are divided into new ones. (b) shows the backbone built from that division, highlighting the intermediate vertices that link different tree levels.



(a) Multi-level clustering (b) Multi-level Neighbor-Joining Tree

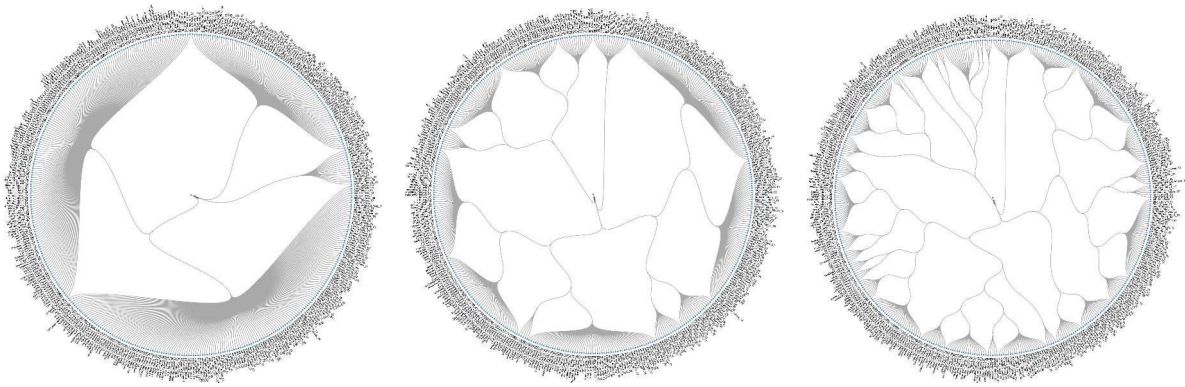
Source: Elaborated by the author.

When compared with the NJ, the multi-level neighbor-joining presents fewer intermediate vertices, being the clustering strategy a powerful way to reduce the space occupied by these vertices. This approach also enables the usage of summarization to visualize larger graphs (in particular with a large number of vertices). A cluster can represent all vertices below itself, which makes it applicable not only for graphs where the number of edges generates the visual clutter but also when it is not adequate to represent all vertices into the visual space. Furthermore, the multi-level execution reduces the computation cost of the entire process, once we replace one execution of the cubic NJ algorithm for the whole dataset for some executions of the same algorithm with small subsets of the original data.

Regarding the parametrization, the user determines the number of clusters, the number of levels and the minimum cluster size. Figure 18 shows three different backbones created with the multi-level neighbor-joining from the dataset INFOVIS15 (ISENBERG et al., 2015) (see section 5.2), when the algorithm is set to produce the multi-level neighbor-joining tree with 1, 2, and 3 levels. By creating different views for the same data, those parameters may improve the user experience with the visualization.

However, those parameters are less accurate to the user, and it is hard to determine a good value for K, which may require a previous knowledge of the dataset. Furthermore, those parameters are a big challenge when developing a proper bundling evaluation and comparison with others techniques. Different from other parameters (discussed later) that can be changed in real-time and preserve the visualization context, any change in the backbone parameters makes a

Figure 18 – Different views of the dataset INFOVIS15 using the multi-level neighbor-joining method with different levels of processing. The increment of levels expands the number of groups, creating a more detailed view, but it also increases the number of intermediate vertices.



(a) 1 level (b) 2 levels (c) 3 levels

Source: Elaborated by the author.

huge impact in the bundling layout. This discussion led us to improve the method to avoid those parameters. We present this method in the next section.

3.2.3 A Cluster-based Backbone Construction

Inspired by the running time and computational cost reduction allowed by the clustering approach on the multi-level neighbor-joining bundling and also looking for a less parametrized technique, we devised a novel strategy for the backbone creation based on the Bisecting k-means (WEISS, 2001). Our goal was to completely avoid the usage of the NJ method, but still produce a similarity tree-like structure to fulfil the constraints imposed by our methodology. To fill this need, we developed a divisive algorithm, called Similarity Tree (STree), to construct a similarity binary tree where the original vertices are sequentially split into clusters and sub-clusters, defining a hierarchy of clusters.

As already mentioned, due to the design principles, the backbone is a tree-like structure where the intermediate vertices V' are internal vertices and the vertices V of G are leaves. Also, to satisfy our primary goal, which is to bundle the edges creating groups according to the dissimilarities amongst the vertices, the length of the path between any two vertices $v_i \in V$ and $v_j \in V$ needs to be proportional to $\delta(d_i; d_j)$. We define this length as the number of intermediate vertices between v_i and v_j. In other words, vertices representing similar objects need to be placed close to each other in the backbone structure.

The Similarity Tree algorithm starts with one cluster C containing all data vertices, representing the root of the backbone. Then, the process splits C into two new clusters C_a and

C_b, so that each cluster contains the most similar vertices among themselves, minimizing

$$\sum_{d_i \in C_a} \delta(d_i; \tilde{C}_a) + \sum_{d_j \in C_b} \delta(d_j; \tilde{C}_b) \tag{3.3}$$

where $\tilde{C}_a$ and $\tilde{C}_b$ represent the centroid of the vertices in C_a and C_b, respectively. After that, C_a and C_b are attached as left and right child of C, and the process is applied to C_a and C_b. This is repeated until singleton clusters are created, that is, clusters containing only one vertex. As a result, the clusters and sub-clusters are intermediate vertices, and the data vertices are leaves. The splitting is based on the same strategy applied by the K-means algorithm (JAIN; DUBES, 1988), with $k = 2$.

The process for the backbone construction is outlined in the Algorithm 1. It receives a dataset D and returns a tree T. In this algorithm, the function PIVOTS(C) is implemented as follows. We first calculate the centroid $\tilde{C}$ of C. Then, we get the fartherst data object $d_a \in C$ from $\tilde{C}$ as the first pivot, and the fartherst object from d_a as the second pivot d_b. Also, the function MEAN(C) returns the centroid of C, that is $\tilde{C} = \frac{1}{|C|} \sum_{d_i \in C} d_i$, where $|C|$ represents the number of objects in C.

Figure 19 shows an example of a tree drawn with the H-Tree algorithm (SHILOACH, 1976). The result of this process is a hierarchical structure in which the deeper the cluster (from the root to the leaves), more similar are the objects belonging to it. As a consequence, the most similar objects are close placed on the backbone, and the path between them is reduced. Also, the resulting tree is a binary tree, with $N - 1$ intermediate nodes. This results on almost the same number of intermediate vertices obtained with the NJ, but in a more balanced tree, that is, a shallower tree. As shown in the next chapter, when we describe the complete evaluation of this method, the Similarity Tree is balanced enough to provide a better edge bundling layout. Furthermore, an intermediate vertex is a (super) cluster representation of its children, so it allows the multiscale analysis over the original tree, by "contracting" the tree vertices at some level.

This section and the previous one presented different tree-like backbone construction strategies developed during this master's research. This step addresses the first stage of our bundling pipeline, consisting of the creation of a structure that will guide the bundling. In the following section, we present the next stage of our bundling pipeline, the vertices placement that will guide the drawing of the curved edges.

3.3 Backbone Placement

Once we have the backbone, the next step is to place the data and intermediate vertices on the plane. The backbone construction method does not provide any spatial information to determine the placement. However, since backbones are tree-like structures, any method for drawing trees can be used, which is normally very fast to be accomplished when compared with

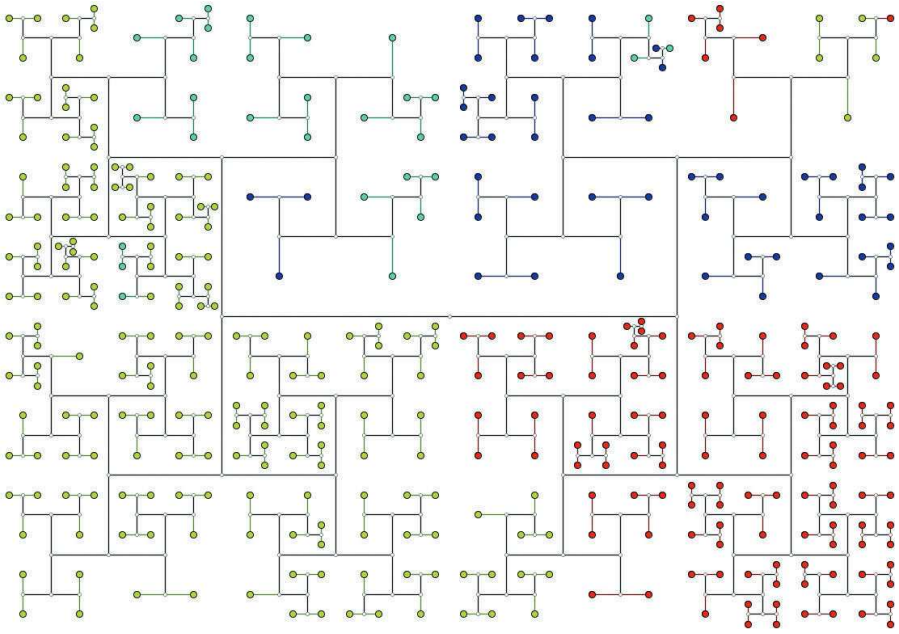
Algorithm 1 Similarity Tree algorithm.

```
function SIMILARITYTREE(D)
  C ← {d_1; ...; d_n} ∈ D                          ◁ assigns all data objects to C
  T.root ← C                                       ◁ creates a tree T and set C as root
  C̃ ← MEAN(C)                                      ◁ sets the centroid of C
  SIMILARITYTREEREC(C, D)
  return T
end function

function SIMILARITYTREEREC(C)
  if |C| > 2 then
    {C_a; C_b} ← SPLIT(C)                          ◁ splits C into two clusters
    C.left ← C_a                                   ◁ sets C_a as the left child of C
    C.right ← C_b                                  ◁ sets C_b as the right child of C
    SIMILARITYTREEREC(C_a)
    SIMILARITYTREEREC(C_b)
  else if |C| = 2 then
    C.left ← C_a                                   ◁ sets C_a as the left child of C
    C.right ← C_b                                  ◁ sets C_b as the right child of C
  else if |C| = 1 then
    C̃ ← d                                          ◁ sets the centroid equal to the single data object
  end if
end function

function SPLIT(C)
  {d_a; d_b} ← PIVOTS(C)                           ◁ selects initial pivots for splitting
  C̃_a ← d_a
  C̃_b ← d_b
  while it < MAX_ITERATIONS do
    for d_i ∈ C do
      if δ(d_i; C̃_a) < δ(d_i; C̃_b) then
        C_a ← C_a ∪ d_i                            ◁ adds d_i to C_a
      else
        C_b ← C_b ∪ d_i                            ◁ adds d_i to C_b
      end if
    end for
    C̃_a ← MEAN(C_a)                                ◁ updates the centroid of C_a
    C̃_b ← MEAN(C_b)                                ◁ updates the centroid of C_b
    it ← it + 1
  end while
  return {C_a; C_b}
end function
```

Figure 19 – Similarity Tree from a synthetic dataset with 380 objects divided into 4 different classes (identified by the vertex color).



Source: Elaborated by the author.

drawing the entire graph. For the neighbor-joining approach, we have performed tests using radial and force-based algorithms. For the Similarity Tree, we took advantage of the binary tree property to devise an adaptation of the H-tree algorithm (SHILOACH, 1976) that preserves the similarity relationships and that makes a better use of the visual space.

The radial layout distributes the vertices on a circle, and places the intermediate vertices over inner concentric circles, with the root on the center. Consequently, the edges only intersect the vertices on their beginning and ending points, thus, avoiding the ambiguity problems related to intersecting vertices and edges, previously discussed. In our strategy, the distribution of the vertices on the circle follows the order imposed by the backbone. The algorithm places the backbone leaves side-by-side following a canonical post-order traversal approach. One important aspect of the radial layout, especially interesting for bundling, is that the focus is on the edges instead of the spatial position of the vertices. Thereby, most of the visual space is used to represent the graph connections, which reduces the visual clutter.

Although we performed some tests using the force-based algorithm with distinct initializations, it is the less suitable layout for bundling purposes. First, because the force layout is an expensive algorithm that takes several iterations to reach a stable state. Second, because it cannot guarantee the absence of crossing paths, even when it is drawing a planar graph (TAMASSIA, 2013). Finally, this method is not stable, which means that different results can be attained for the same input in multiple iterations.

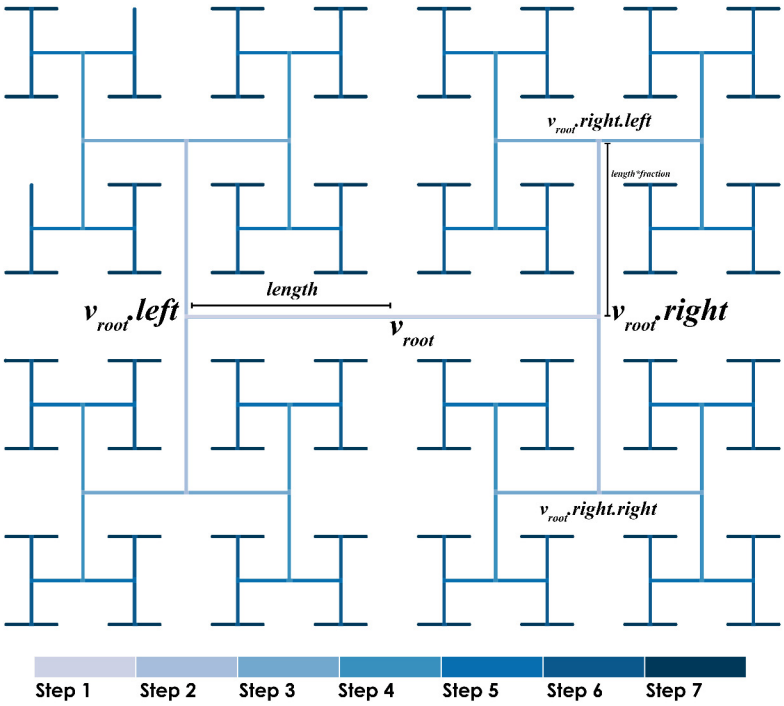
On the other hand, the H-tree layout focuses on the vertices. The H-tree algorithm places the vertices in a recursive process linking nodes through perpendicular line segments, resulting in

a fractal structure with a repeating pattern that resembles the letter "H". By equally dividing the available space to represent the two sub-trees of each node, the resulting layout effectively uses the available visual space on the presence of balanced trees. However, for unbalanced trees, it fails because the space is split into two half-parts independently of how many vertices belongs to each sub-tree. In the following sub-section, we detail how we adapt the H-tree algorithm to make a better use of the available space when handling unbalanced trees and to preserve similarity relationships.

3.3.1 Swapping H-tree

On the original H-tree, the vertices are positioned with a recursive process that starts placing the root v_{root} of the tree on the center of the visual space and its two children $v_{root}.left$ and $v_{root}.right$ (if they exist) horizontally equidistant from v_{root} by a $length$. Then, the children of $v_{root}.left$ and $v_{root}.right$ are positioned vertically equidistant from $v_{root}.left$ and $v_{root}.right$, respectively, by a fraction of $length$. This process is repeated alternating the direction (vertical/horizontal) and reducing $length$ by a factor of $\sqrt{2}$. Figure 20 shows how the layout is built after each recursive step of this algorithm.

Figure 20 – An H-Tree layout represented by each iterative step. In this example, each edge is drawn with the color that indicates which step the vertices were processed.



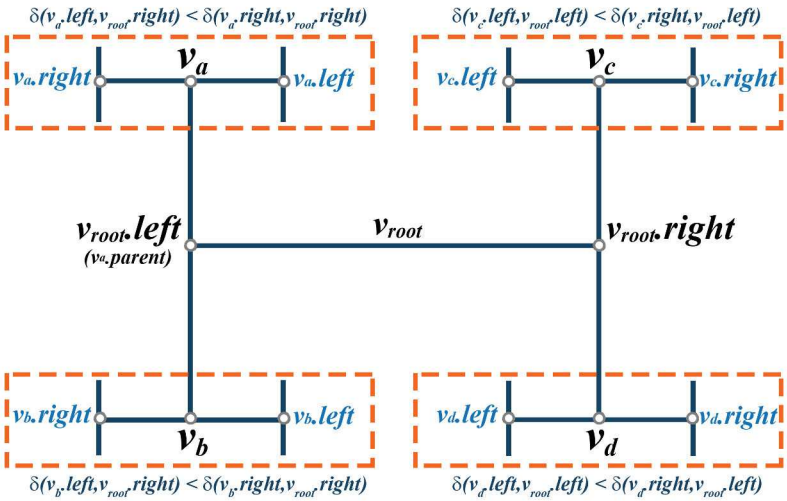
Source: Elaborated by the author.

Since our goal is to preserve distances, we modified this algorithm to allow swaps between siblings vertices, that is, a right sibling becomes a left sibling and vice-versa. Figure 21 explains this modification. Considering that the vertex v_a has been processed, with coordinates

on the plane assigned to it, before positioning its children, $v_a.left$ and $v_a.right$, we perform a test to check if swapping them will improve the distance preservation.

Once $v_a.left$ and $v_a.right$ represent different clusters of objects (their centroids), and that $v_{root}.right$ represents a (super) cluster encompassing different clusters ($v_c.left$, $v_c.right$, $v_d.left$, and $v_d.right$), we can swap $v_a.left$ with $v_a.right$ in order to place close to $v_{root}.right$ the most similar cluster (vertex) to it, if it improves the distance preservation between the clusters represented by $v_a.left$ and $v_a.right$ with respect to $v_{root}.right$. The same applies to $v_b.left$ and $v_b.right$ with respect to $v_{root}.right$, $v_c.left$ and $v_c.right$ with respect to $v_{root}.left$, and $v_d.left$ and $v_d.right$ with respect to $v_{root}.left$.

Figure 21 – Swapping branches using the H-Tree layout. The image shows, for v_a,v_b,v_c and v_d, the dissimilarity evaluation made to decide which branch is positioned in the center of the visualization, and which one is positioned outside.



Source: Elaborated by the author.

Through those swaps, it is possible to improve the distance preservation between the vertices belonging to neighbors (super) clusters, represented by $v_{root}.left$ and $v_{root}.left$, since it will place the most similar vertices close on the final layout. These swaps potentially lead to the improvement of the overall distance preservation of the produced layout without corrupting the tree topology since only swaps between siblings are allowed.

Notice that this swap strategy performs only local modifications, so it does not guarantee that we are producing the H-tree arrangement that best preserves the distance relationships on the plane. Nevertheless, due to the triangle inequality axiom of distance functions, we expect to improve (when possible) the results of any given tree. The complete process is detailed in Algorithm 2. In this algorithm, the function SWAPSIBLINGS($v.left; v.right$) simply swaps the sibling nodes $v.left$ and $v.right$, and $\tilde{v}_i$ represents the centroid of the cluster represented by v_i.

Different from the radial layout, the H-tree layout focuses on the vertices instead of on

Algorithm 2 Swapping H-tree algorithm.

```
function SWAPPINGHTREE(T)
  SWAPPINGHTREEREC(T.root, initial_length, false)
  return T
end function

function SWAPPINGHTREEREC(v, length, horizontal)
  if v == T.root then                          ◁ Set the root vertex to the layout's center
    v̂.x ← 0
    v̂.y ← 0
  else
    if horizontal == true then                 ◁ If it is an horizontal placement
      if v is left child then
        v̂.x ← v̂.parent.x − length
      else
        v̂.x ← v̂.parent.x + length
      end if
      v̂.y ← v̂.parent.y
    else                                       ◁ If it is an vertical placement
      if v is left child then
        v̂.y ← v̂.parent.y + length
      else
        v̂.y ← v̂.parent.y − length
      end if
      v̂.x ← v̂.parent.x
    end if
    SWAP(v)                                    ◁ Swap the children of v if it improves the layout
    SWAPPINGHTREEREC(v.left; length = length/√2; !horizontal)
    SWAPPINGHTREEREC(v.right; length = length/√2; !horizontal)
  end if
end function

function SWAP(v)
  if v.parent == v.parent.parent.right then
    v_P ← v.parent.parent.left
    v_N ← v.left
    v_F ← v.right
  else
    v_P ← v.parent.parent.right
    v_N ← v.right
    v_F ← v.left
  end if
  if δ(ṽ_F; ṽ_P) < δ(ṽ_N; ṽ_P) then
    SWAPSIBLINGS(v.right.left, v.right.right)
  end if
end function
```

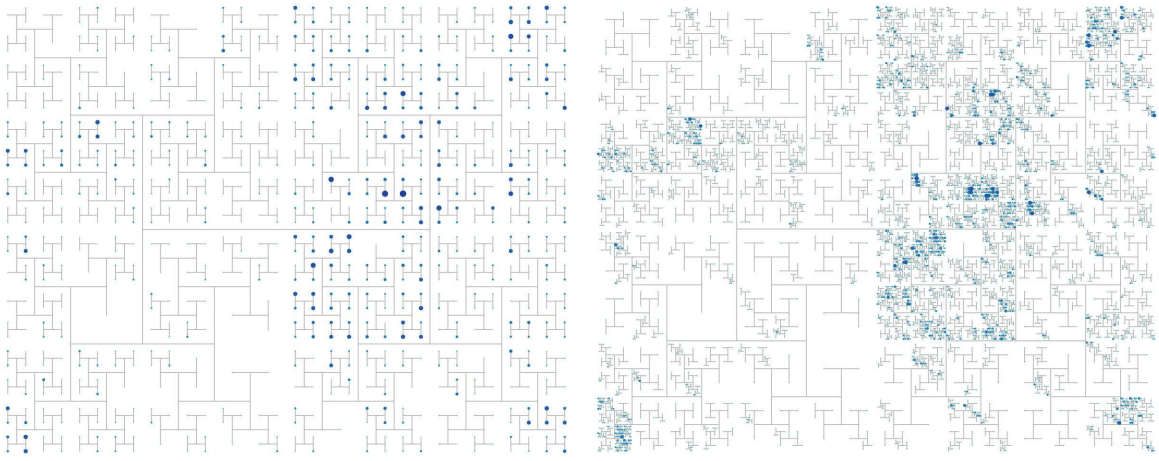
the edges. Therefore, the distances between the data objects are better preserved. However, it is expected to generate more cluttered layouts due to the reduced space for the edges. Also, the radial layout provides a clear separation between the space designed for leaves (around the circumference) and intermediary nodes and branches, that are overlapped by the bundling edges. The H-tree layout places the vertices spread on the plane, creating some overlap with edges.

3.3.2 Multiscale Backbone Placement

A important question during the backbone placement is how to handle the scalability of the layout. The circular layout organizes all vertices over a circumference, which limits the layout by the length of this circumference. Therefore, the circular layout is not useful for large datasets. On other hand, the H-Tree layout supports more vertices, since it makes a better use of the visual space. However, because the algorithm divides the length of each branch (and thus its available area), the layout is also limited in deeper levels, where the small drawing area makes the graph visualization unreadable.

In order to reduce the impact of this limitation, we added a parameter, called τ, that control the maximum depth in which vertices will be placed in the visualization. When the vertices placement algorithm reaches this level, the intermediate vertex that represents its cluster is defined as the final vertex and the algorithm stops the recursive process. Thereby, all vertices of this branch will be hidden. Although this process reduces the available information, the user can control the value of τ, thus visualizing the graph in multiple levels. Figure 22 shows a comparison of two values of τ for a huge graph. When $\tau = 10$ all vertices are easily detected and the size of each circle represents the number of hidden vertices that are below. For larger values of τ, more vertices are displayed, but they are positioned close in deeper levels, which makes them hard to be identified.

Figure 22 – Comparison between the backbone placement of a huge graph using different values of τ.



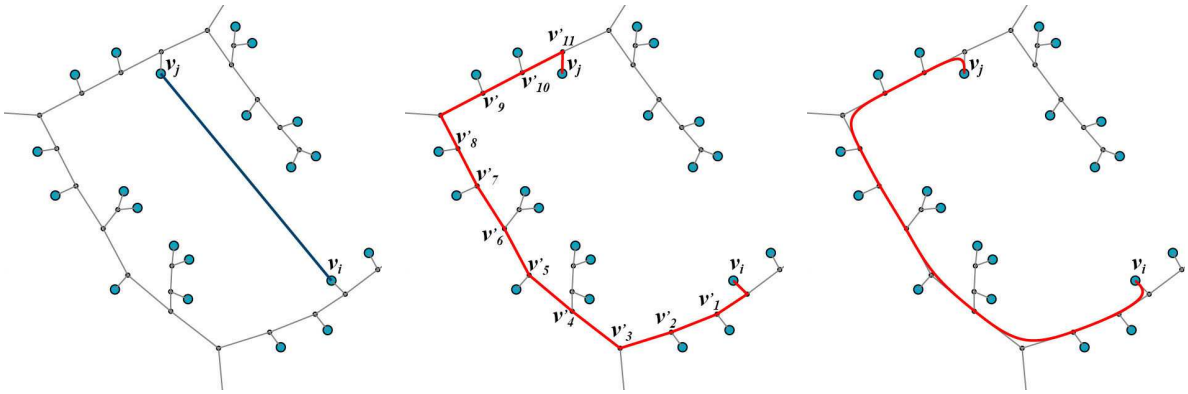
(a) $\tau = 10$ (b) $\tau = 15$

Source: Elaborated by the author.

3.4 Edges Drawing and Bundling Enhancements

Finally, the last task for drawing the graph is to bend the edges in order to compose edge bundles. In this process, for each edge e_{ij} in G we first search the path $p_{ij} = \{v_i; v'_1; v'_2; \ldots; v_j\}$ on the backbone that connects v_i and v_j. Then, we bend e_{ij} towards p_{ij} by considering its vertices as control points of a B-Spline curve. Since we can guarantee that the hierarchy has one path connecting any pair of original vertices over a subset of virtual vertices (design principle (1)), we first find those paths executing a simple breadth-first search from the root of the tree. By knowing the path from the root to every original vertex, we can find any path between two original vertices in a faster and simpler way than any shortest-path algorithm for graphs. An overview of this process is shown in Figure 23.

Figure 23 – Curving an edge through the backbone. First, the original edge is selected (a); the path between the source and the target is found (b); a B-Spline curve is drawn using the intermediary vertices as control points.



(a) Step 1 (b) Step 2 (c) Step 3

Source: Adapted from Sikansi and Paulovich (2015).

Regarding the multiscale placement, the edge drawing is modified according to the available vertices. When part of the vertices in the path p_{ij} is hidden, the edge is rendered using only the available vertices. If the path p_{ij} contains only hidden vertices the edge is not rendered. Thereby, the multiscale placement filters local edges, hence it controls the amount of visible edges in favor of edges connecting less similar vertices.

3.4.1 Control Points Filtering

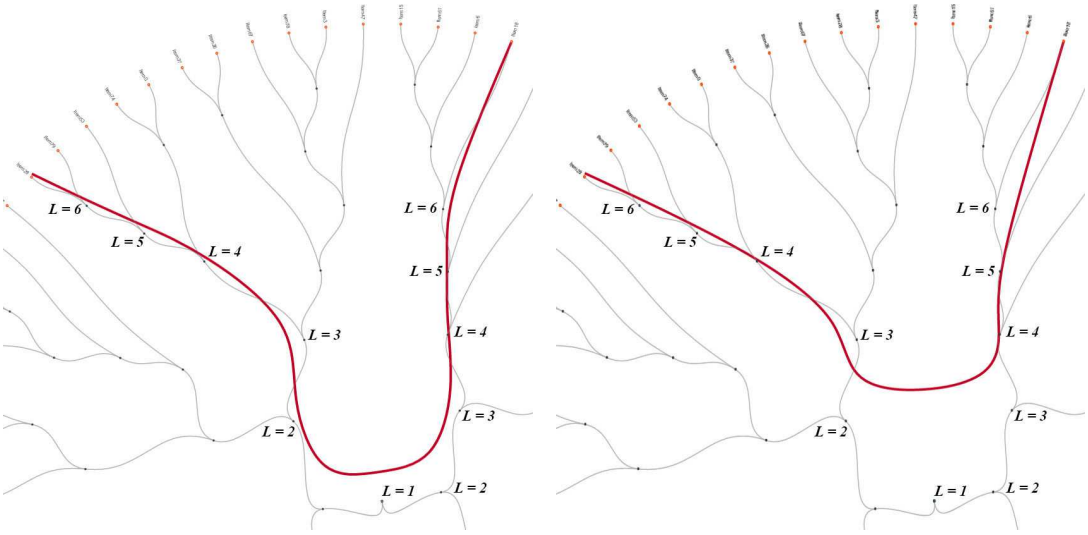
The standard edge drawing uses all intermediate vertices in the path p_{ij} to bend the edges. However, bundles can be enhanced by filtering the intermediate vertices used as control points. To perform this filtration, a function $\Phi(v'_k)$ establishes if an intermediate vertex should be used as a control point, and the edge is curved towards the path $p'_{ij} = \{v_i; v'_k \in I \mid \Phi(v'_k); v_j\} \subset p_{ij}$, being I the set of intermediate vertices in the path p_{ij}. Although there are many ways to determine $\Phi(v'_k)$, we defined it in function of the level $L_{v'_k}$ of each intermediate vertex v'_k. $L_{v'_k}$ represents the

stage in which the given cluster was divided and, consequently, the distance from the vertex to the root v_{root}. Thus, $L_{v_{root}} = 1$ because it is defined in the first iteration of our algorithm. $\Phi(v_k)$ is defined as follows:

$$\Phi(v'_k) = \begin{cases} \text{true} & \text{if } \varphi_L \leq L_{v'_k} \leq \varphi_U; \\ \text{false} & \text{otherwise.} \end{cases} \tag{3.4}$$

where $\varphi_L \leq \varphi_U$. The parameters φ_L and φ_U are controlled by the user. These values work as a lower and upper delimiter to decide if the intermediate vertex v_k will be used as control point. φ_L determines how far from the root the intermediate vertex v_k should be, while φ_U determines the maximum level used to bend the edges. Figure 24 shows a comparison between an edge bent towards p_{ij} and p'_{ij}. Filtering intermediate points avoid the construction of a bundling layout tightly close to the backbone. The lower bound improves the identification of connections among opposite groups, while the upper bound reduces the distortion caused by unbalanced branches.

Figure 24 – Comparison between a curved edge when no filtration is performed (a) and a edge drawn with control points in the interval [4;6] (b).



(a) $\Phi(v_k) = \text{true}$ (b) $\varphi_L = 4$ & $\varphi_U = 6$

Source: Elaborated by the author.

3.4.2 Adaptive-β

In addition to the intermediate vertices filtration, the user can also control the edges' curvatures by moving the control points in the path p'_{ij} towards the straight line connecting v_i and v_j. This enhancement is similar to the approach presented by Holten (2006). However, instead of considering a constant value for all edges, we calculate different values for each edge depending on its size. The main idea is that a edge bundling layout tends to hide short edges while highlight

bundles formed by long ones. Therefore, we changed this process to remove short edges from bundles. Given the position $\hat{v}_k$ of the vertex $v_k \in p'_{ij}$, we calculate the transformed position as

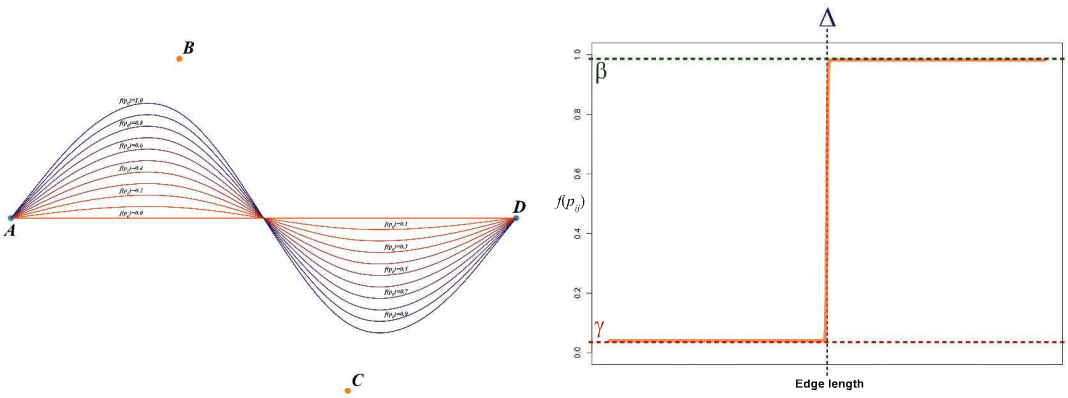
$$\hat{v}_k = f(p'_{ij})\hat{v}_k + (1 - f(p'_{ij}))(\hat{v}_i + \frac{k}{|p'_{ij}|}(\hat{v}_j - \hat{v}_i)) \tag{3.5}$$

where $\hat{v}_i$ and $\hat{v}_j$ are the transformed positions of the ending points of an edge e_{ij}, k is the index of v_k in p_{ij}, and $f(p_{ij})$ is a function that controls the bending, given by

$$f(p_{ij}) = \frac{\beta - \gamma}{1 + e^{\frac{\Delta - d(\hat{v}_i;\hat{v}_j)}{0.05}}} + \gamma \tag{3.6}$$

$f(p_{ij})$ ranges from [0;1] to allow a proper convex interpolation between the straight line and the B-spline curve. The function $f(p_{ij})$ has three parameters defined by the user: Δ is the minimum size for edges that will be bundled, β is the strength of curvature for bundled edges and γ is the strength of curvature for unbundled edges. In addition, $d(\hat{v}_i;\hat{v}_j)$ is the original edge length (Euclidean distance). Usually, β is set with a value close to 1, while γ is set with a value close to 0. Basically, this manipulation creates a function with a hard-step that works as an activation function (its behavior approximates a step-function). This function consider the size of each edge and determine if it should be placed in a bundle or draw straight to its target. Figure 25 shows the effect of this interpolation in a curve and the behavior of this function according to the parameters used.

Figure 25 – The interpolation between the straight and the curved edge. (a) shows how the value $f(p_{ij})$ affects the drawing of an edge connecting A and D through the control points B and C. The curved lines show an interpolation from the straight line ($f(p_{ij}) = 0.0$) up to the most distorted drawing ($f(p_{ij}) = 1$). (b) shows a plot of $f(p_{ij})$ according to the parameters β, α and Δ, set by the user. Δ determines the minimum size of an edge to be placed in a bundle, β is the strength of curvature for bundled edges (usually set close to 1.0) and γ is the strength of curvature for unbundled edges (usually set close to 0.0).

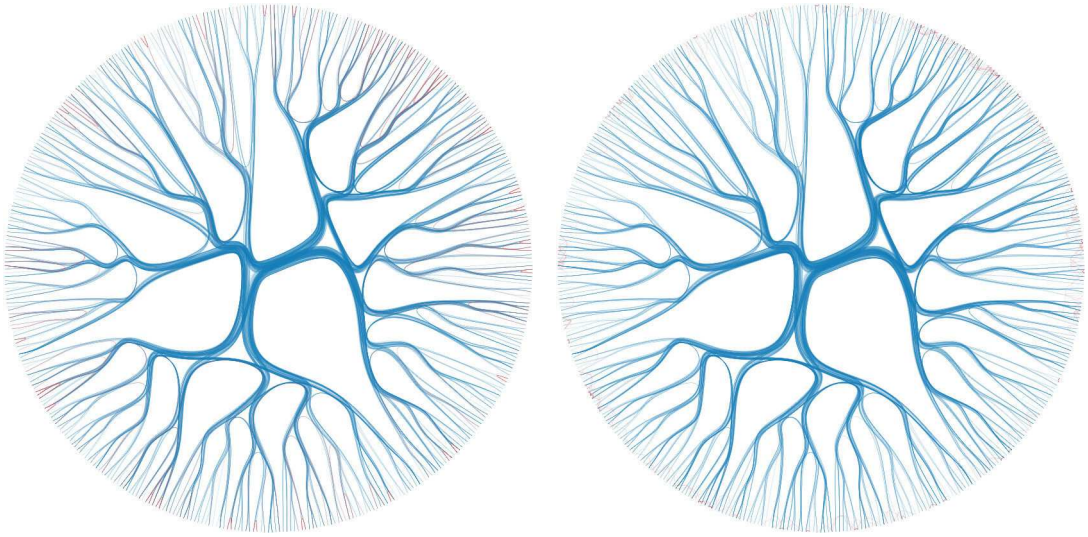


(a) Different curves according to $f(p_{ij})$ (b) $f(p_{ij})$

Source: Elaborated by the author.

Because there are no conclusive experiments about the best values for the strength of curvature, it is useful to allow users to change this parameter interactively. A small value of $f(p_{ij})$ produces a less distorted graph, but with less clutter reduction. On the other hand, $f(p_{ij})$ values close to 1 produce the best result concerning the clutter reduction, but with more distortion. Therefore, $f(p_{ij})$ will result in two different scenarios. Edges with $d(\hat{v}_i;\hat{v}_j) < \Delta$ will have their curvature strength equal to γ, while edges with $d(\hat{v}_i;\hat{v}_j) > \Delta$ will have their curvature strength equal β. Once γ is close to zero, those edges will not be placed in bundles and they will be rendered similarly to straight lines. We believe that small edges should not be included in bundles, since they generate too much distortion and ambiguity, depending on the dataset. However, removing edges from bundles increases the visual clutter, so the parameter Δ is used to balance this trade-off. Figure 26 shows a comparison between the original formulation and our proposal.

Figure 26 – Comparison between β proposed by Holten (2006) and our proposal of an adaptive-β using the dataset INFOVIS15. The comparison shows short edges (smaller than a length threshold) in red, those edges are better identified in our proposal. For this visualization we used the values $\beta = 0.97$, $\gamma = 0.1$ and $\Delta = 0.2$.



(a) Holten (2006) β' variation (b) Our β' variation

Source: Elaborated by the author.

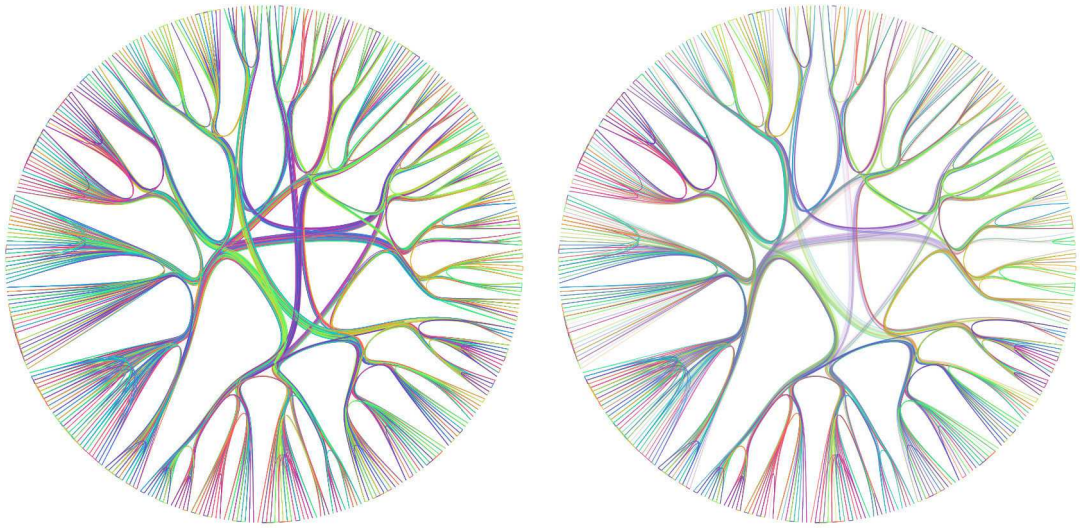
3.4.3 Opacity and Coloring

Finally, opacity and coloring can improve the bundling appearance. There are many examples of their usage in prior edge bundling applications. For example, a color scale can be used to represent an attribute or aspects of each edge. Telea and Ersoy (2010) and Selassie, Heller and Heer (2011) use colors to represent bundles or clusters, in order to improve the identification of distinct groups of edges. Holten (2006) uses a gradient to represent each edge direction, with different colors that identify the source and the target of an edge. This approach is useful since other solutions, such as arrows, are unlikely to work in a cluttered layout. However, our

experiments showed that this method loses efficiency when it needs to face graphs with more complex edge patterns, such as bundles that contain a huge number of edges in both directions.

In addition to the color, manipulating the alpha channel may be helpful to highlight the level of aggregation in each bundle. More specifically, opacity has a crucial role in representing the number of edges grouped in a bundle. Once the bundle is formed by an overlap of edges, the combination of multiple layers increments the bundle intensity. Thus, bundles with more edges have a strong intensity, while bundles with fewer ones are less recognizable. This value also hides non-bundled edges, which will have the less intensity in the layout. The bundling layout can have a global opacity, which can be modified in real time, or a different value for each edge, as presented by Holten (2006) to highlight short edges. Figure 27 shows an example of bundling colorization with and without a global opacity. The edge color is determined by the angle formed by the edge and the horizontal axis. When only the colorization is used, all bundles seem to have the same intensity, while reducing the global opacity reveals the most and the least representative bundles.

Figure 27 – Comparison between a colorized bundling layout and the combination of color and opacity.



(a) Only color (b) Color and opacity

Source: Elaborated by the author.

3.5 Final Remarks

In this chapter, we presented the main formulation of this research work. During the research, from the neighbor joining to the similarity tree, each step represented a movement toward of the main goal. This technique was beyond to the state-of-the-art presented before, once, although the aspect that similarity representation is a prominent discussion on edge bundling related publications, none of them have built an entire similarity-based approach before.

CHAPTER

4

RESULTS

4.1 Initial Remarks

In this chapter, we report and evaluate the results of the Similarity Bundling Framework. Regarding edge bundling evaluation, prior techniques often concentrate their reports on applications and sometimes perform informal user studies. For example, Holten (2006) presented user studies that recognized the effectiveness of an edge bundling layout for quickly gaining insight into the relationship among elements. However, most techniques limited their discussion to the computational cost and applications, without an extensive evaluation.

Thereby, a remaining challenge is the lack of an effective evaluation of those methods according to their all features, not only their computational costs. This problem is mentioned in many publications (ERSOY et al., 2011; HURTER; ERSOY; TELEA, 2012; NGUYEN; EADES; HONG, 2013a), showing that there is no consensus about an evaluation framework that compares different methods. Therefore, we developed an evaluation pipeline that verifies the ability of our backbone, the Similarity Tree, to express similarities relationships and the quality of the Similarity-driven Edge Bundling layout.

Our evaluation pipeline is divided into two sections. First, we performed a comparative analysis of the Similarity Tree with others well-known tree construction methods (Neighbor-Joining and UPGMA). In this analysis, two metrics were used to compare those methods in terms of similarity representation and tree balance. In addition, we validated the swapping transformation designed for the H-Tree layout. Second, we performed an analytical evaluation of our method, comparing it with different edge bundling techniques, namely CUBu, MINGLE and FDEB. We initially present a short analysis using the Neighbor-Joining and its multi-level transformation to create the backbone. To differentiate these methods from that using the Similarity Tree, we called them Neighbor-Joining Bundling (NJB) and Multi-level Neighbor-Joining Bundling (Multi-level NJB), respectively. Next, we describe the datasets employed in

this comparative analysis.

4.2 Datasets

Our experiments and comparative studies used public datasets from previous publications, as well as artificial datasets generated during this research. Those datasets were chosen due to their specific properties that allowed us to evaluate each stage of our method according to previous knowledge about the data. This section describes the datasets used for the backbone evaluation and the ones used to compare the edge bundling layouts created with the Multi-level Neighbor-Joining and the Similarity Tree.

4.2.1 Datasets for the Similarity Tree Evaluation

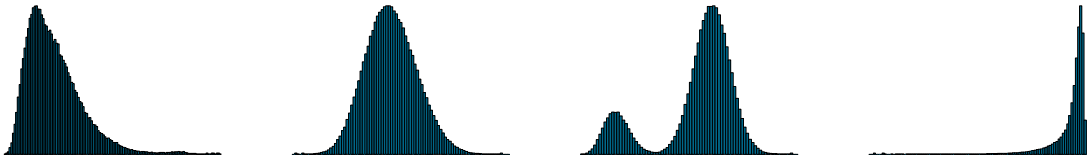
To evaluate the backbone construction, we used four datasets from a previous study on visual representation of multidimensional data (FADEL et al., 2015). These datasets represents different data distributions and were already used to represent problems often detected in such analysis. The four datasets are described below and their distance distributions are presented in Figure 28.

- WDBC: a breast cancer dataset obtained from digitized images of breast masses (ASUNCION; NEWMAN, 2007) classified according to two classes. In this dataset, most data objects are similar among themselves, with few dissimilar ones.
- TWO_NORM: an artificial dataset with two classes, composed of multidimensional points from two Gaussian distributions with unit covariance matrix. There is a good distribution between similar and dissimilar objects, composing well-separated groups of data objects.
- SIMPLEX: an artificial dataset with six well-separated classes. This dataset consists of m-dimensional spherical Gaussian points with a predefined standard deviation and means at the corners of a m-dimensional simplex.
- TEXT: a vector space model representation of scientific papers from four distinct areas. In this dataset, most data objects are dissimilar among themselves, with few similar objects, a feature normally found on high-dimensional sparse spaces.

4.2.2 Datasets for the Multi-level NJ Bundling Evaluation

Our bundling method was evaluated using artificial undirected graphs. Recalling that our bundling framework uses information from data objects to create the backbone and edges are rendered following this structure, our synthetic sets of vertices and edges were generated in different ways aiming at testing different properties. Moreover, this allowed us to apply different

Figure 28 – Distance distributions of the four datasets employed on the backbone evaluation.



(a) WDBC (b) TWO_NORM (c) SIMPLEX (d) TEXT

Source: Elaborated by the author.

sets of edges to the same set of vertices. We call DATASET each set of vertices and its related data that is used to construct the backbone, and DISTRIBUTION each set of edges.

The results obtained with the Multi-level Neighbor-Joining are presented using three graphs. Those were generated considering different edges distributions from the same DATASET of 600 data objects. These elements are equally divided into five well separated clusters (C_1;C_2;C_3;C_4;C_5). Each DISTRIBUTION contains 4,000 edges, where a few edges connect randomly any pair of elements (noise) and the remaining ones follow one specific pattern. The three distributions are presented below:

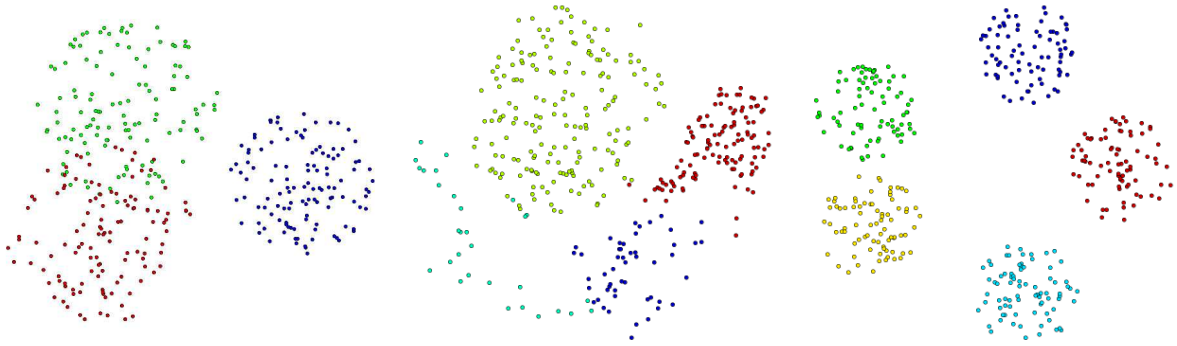
- DISTRIBUTION 1: apart from noise, most edges connect elements from clusters C_1 and C_3 and elements from clusters C_2 and C_5;
- DISTRIBUTION 2: apart from noise, most edges connect elements from clusters C_1 and C_4 and elements from clusters C_2 and C_4;
- DISTRIBUTION 3: apart from noise, all the other edges connect elements inside cluster C_3 and inside cluster C_5;

4.2.3 Datasets for the Similarity-driven Edge Bundling Evaluation

The evaluation of the Similarity-driven Edge Bundling used six graphs also generated with artificial data. In this case, we combine different DATASETS and edges DISTRIBUTIONS. We created three set of vertices following different similarity relationships, namely DATASET I, DATASET II and DATASET III. The differences among them include the number of elements, classes and intersections. Figure 29 shows a two-dimensional representation of the three artificial datasets.

To create the six different graphs, for each set of vertices, we defined two edges distributions, namely DISTRIBUTION A and DISTRIBUTION B. The set of edges were planned to show a pattern or relationship among elements considering their classes. These patterns can be visualized in a matrix of adjacency, and we can compare the bundling result with the adjacency visualization. Table 1 summarizes the artificial datasets and presents a matrix visualization of

Figure 29 – Two-dimensional representation of the three artificial datasets used in the Similarity-driven Edge Bundling evaluation.

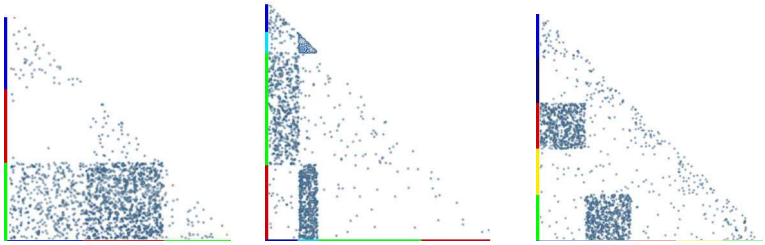
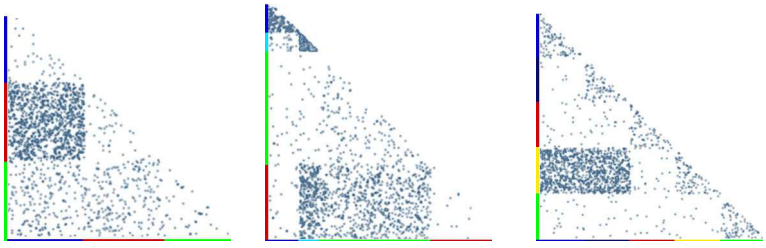


(a) Dataset I (b) Dataset II (c) Dataset III

Source: Elaborated by the author.

each edge distribution. All sets of edges have 1,600 elements. We defined both size of vertices and edges following the most common size used in former edge bundling papers.

Table 1 – Summarization of the artificial datasets used in the Similarity-driven Edge Bundling evaluation.

	DATASET I	DATASET II	DATASET III
Number of classes:	3	4	5
Elements per class:	[120; 120; 120]	[50; 30; 180; 120]	[80; 80; 80; 80; 80]
DISTRIBUTION A			
DISTRIBUTION B			

The following sections describe our evaluation process, starting with the evaluation of our backbone, followed by our bundling evaluation.

4.3 Backbone evaluation

We chose two measures to evaluate the Similarity Tree (STree). The first one is the neighborhood preservation, which evaluates the ability to represent the similarity relationships

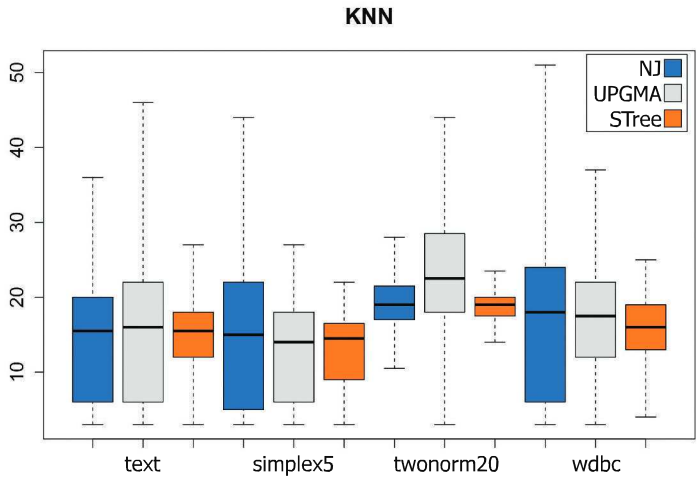
of a given dataset. The last one is the average path size from root to leaves, which evaluates the capability to create less distorted branches in the resulting backbone. We compared our method against other techniques that present the same goal of distance preservation: the Neighbour-Joining (SAITOU; NEI, 1987) and the UPGMA hierarchical clustering (SOKAL; MICHENER, 1958). In the first test, we evaluated the neighborhood preservation of the original space conveyed by the three different techniques. In other words, objects from the same neighborhood should be placed nearby on the tree. We calculated the distance preservation of a data object d_i as follows. First, we compute the k-nearest neighbours of d_i, resulting on a list $NN_i = \{d_{i_1}; d_{i_2}; \ldots; d_{i_k}\} \subset D$. Then, for each element $d_{i_j} \in NN_i$ we computed how many nodes are between the node representing d_i and the node representing d_{i_j}, i.e., the length of the path. The distance preservation of d_i is the median of these values, with small values indicating better results. In this way, we favor the local preservation, that is, well-constructed trees are the ones that closely link the most similar objects and, consequently, place the neighbor objects closely when embedded on the plane.

Figure 30 presents boxplots of the results comparing the methods for each dataset. The blue boxplots represent the results conveyed by the NJ; the gray boxplots depict the results of the UPGMA; and the orange ones outline the results of our approach, the STree. To obtain these boxplots, we varied the neighborhood size from 5 to 20 and computed the preservation distance for each data object. On average, the results produced by the STree are very close or better than the results of the others, with a smaller deviation from the average. Thereby, the layout produced by the STree is more "reliable" since the degree of neighborhood preservation is uniformly distributed over the entire tree, without bad spots that are compensated on the average by good spots. It is worth noting that NJ and UPGMA are more computationally expensive techniques than STree.

Regarding tree balance, the STree presented better results when compared to the other techniques. Figure 31 presents the results of each technique for each dataset. Again, the blue boxplots represent the results for the NJ algorithm, the gray ones for the UPGMA, and the orange ones for the STree. These boxplots summarize the depth of the leaves. The red line represents the results of a perfectly balanced tree. We divide the depth of each leaf by $\log_2 n$, where n accounts for the number of data objects.

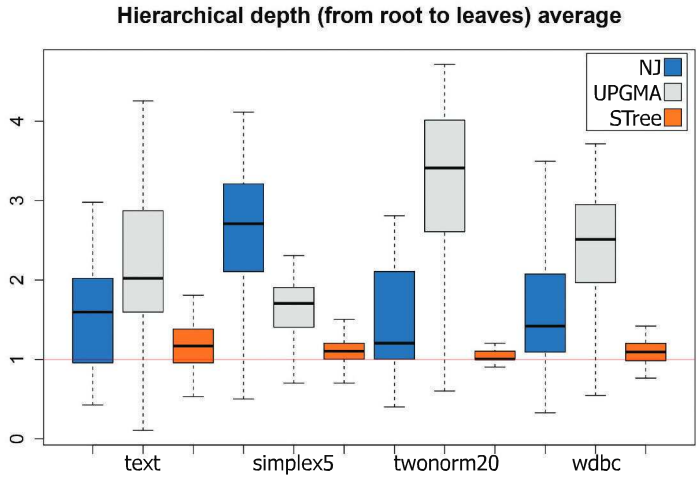
On average, the results rendered by the STree are very close to the red line, while other techniques deviate from it, indicating that the STree creates more balanced trees. On balanced trees, the positive side is that less intermediate nodes (on average) are between one data object to any other data object (leaves). Thereby, resulting in less distorted edges since less bending is introduced in the process. Moreover, this provides a more consistent use of the available visual space, especially for the H-tree layout, reducing the visual clutter resulted from the edges overlapping. On the radial layout, the uniform depth of STree results on a uniform distortion of the edges, avoiding long distorted edges that could be produced by the NJ or UPGMA trees.

Figure 30 – Boxplots summarizing the neighbourhood preservation attained by the STree, NJ, and UPGMA considering the four distinct datasets. On average, the STree presents very competitive results if compared to more expensive techniques, with a smaller deviation. Therefore presenting less bad spots in terms of neighborhood preservation.



Source: Elaborated by the author.

Figure 31 – Boxplots summarizing the balance attained by the STree, NJ, and UPGMA considering the four distinct datasets. The STree produces more balanced trees, therefore leading to less distorted edges on the bundling and a better use of the available visual space. The red line indicates the perfect balance.



Source: Elaborated by the author.

4.3.1 Swapping Evaluation

We also evaluated whether our H-tree swap strategy effectively improves the distance preservation on the produced layouts or not. We defined as distance preservation the degree of how much the pairwise distance between the data objects is preserved in the visual space considering their corresponding leaves' positions. The most common approach for such kind of

evaluation is a metric stress function, such as the Kruskal stress (KRUSKAL, 1964). However, the metric evaluation is not applicable because we do not use distances to position the leaves, we only use the topology of the tree. Hence, the distance between two leaves are not directly proportional to the distance of their corresponding data objects in the original space.

We opted to use a non-metric evaluation based on the rank of the distances. The distance preservation of a data object $d_i \in D$ is computed as follows. We calculate a rank $R_{d_i} = \{r_{d_1}; r_{d_2}; \ldots; r_{d_n}\}$ comparing the data objects in D with d_i, assigning 1 to the most similar and n to the least similar data object, where n accounts for the number of data objects. For the corresponding leaf v_i, we calculate another rank $R_{v_i} = \{r_{v_1}; r_{v_2}; \ldots; r_{v_n}\}$ comparing its position on the plane with the positions of the other leaves, assigning 1 to the closest leaf and n to the one farthest positioned. The distance preservation is then computed comparing both ranks using the Spearman rank-order correlation coefficient (SPEARMAN, 1904), given by

$$r_s = 1 - \frac{6 \times \sum_i^n (r_{d_i} - r_{v_i})^2}{n^3 - n} \tag{4.1}$$

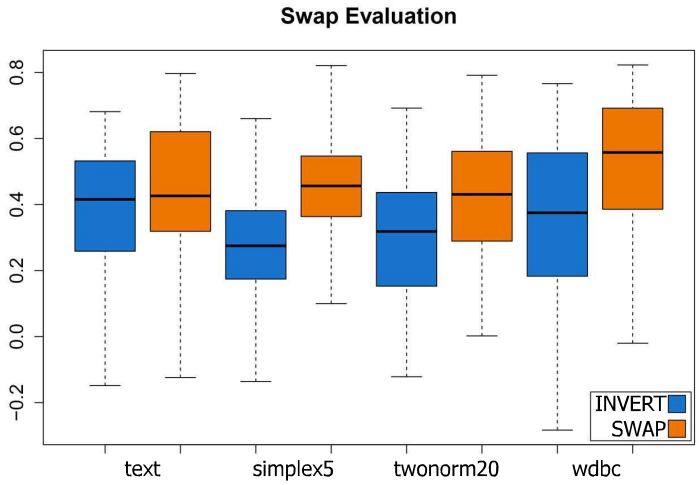
r_s varies in the range of $[-1; +1]$, with larger values indicating better rank-order preservation or, in our case, distance preservation.

We compared our approach with the opposite swap, that, which is expected to produce worst results. Because our approach only performs local changes, instead of global ones, it is possible that random swaps might produce better results. In our tests, none random swapping leads to layouts that better preserves distance relationships, but we cannot guarantee that it will never happen.

Figure 32 presents boxplots summarizing the results of our swapping strategy for each dataset. The blue boxplots represent the opposite swaps and the orange boxplots represent our strategy swaps. For all datasets, the average distance preservation attained by the normal swap is considerably better than the inverted swap. Only for the TEXT dataset this improvement is not evident. Since in this dataset the distance distribution indicates that most data objects are dissimilar among themselves (see Figure 28), without a clear dissimilarity "ranking" amongst them, this is an expected outcome. In fact, such kind of distance distribution is usually observed on high-dimensional sparse spaces, and it is known that distance metrics are poorly-defined for such spaces.

The experiments above validated the Similarity Tree as a reliable process to generate the backbone, which is considered the first part of our methodology. The Similarity Tree is a process faster than conventional similarity-driven tree construction algorithms, and produces more balanced trees. Therefore, this process is very competitive with other listed techniques to represent the similarity among elements, and it was further improved with our swap strategy.

Figure 32 – Boxplots summarising the results, in terms of distance preservation (rank), of the swap strategy. If compared to the opposite swap, the results are considerably better, showing its efficacy on improving the distance preservation



Source: Elaborated by the author.

4.4 Bundling Evaluation

A few metrics have been proposed to assure the quality of an edge bundling layout. For example, Gansner et al. (2011) calculated the amount of ink saved to measure the clutter reduction, but did not address the graph readability. Another metric is the bundling stress proposed by Nguyen, Eades and Hong (2013b). Based on the Kruskal stress, this metric aims to measure the difference between edges compatibilities and their distance in the bundling layout. Although this metric seems promising, we could not determine a trustworthy way to compute the compatibility among edges.

Therefore, we compared our technique with others from the state-of-the-art in a qualitative analysis. Specifically, we chose one method from each group of techniques presented in chapter 2. From the group of image-based techniques, we used the technique CUBu. The techniques FDEB and MINGLE were selected to represent the force-based and the geometry-based techniques, respectively. In all experiments, we generated CUBu layouts using the source code provided by the authors, while we generated the graphs from MINGLE[1] and FDEB[2] using open source libraries.

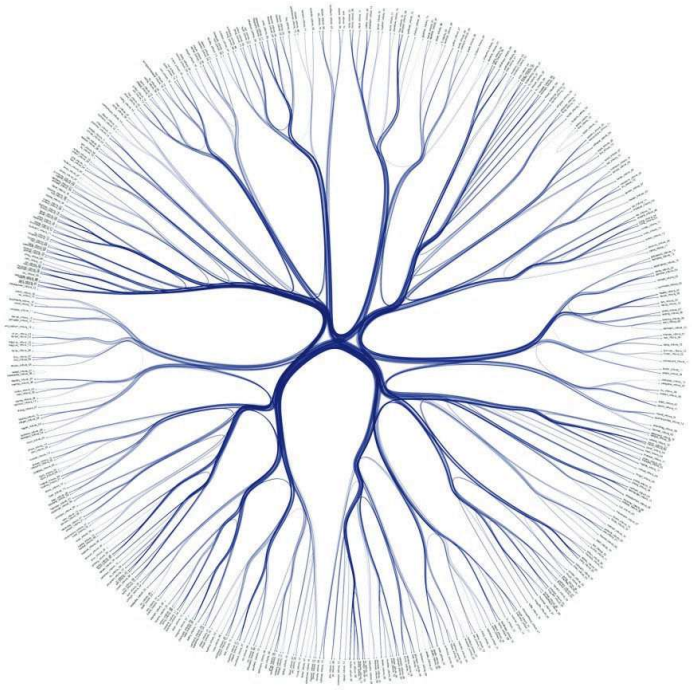
4.4.1 Neighbor-Joining Bundling

The first visualization devised from our framework uses the Neighbor-Joining to create the backbone and the Reingold-Tilford radial layout to place the vertices in the visual space. We

1 https://github.com/philogb/mingle
2 https://github.com/upphiminn/d3.ForceBundle

applied these methods to visualize the graph of citations from a subset of the 2015 IEEE Infovis dataset (ISENBERG et al., 2015), comprising papers published on the IEEE InfoVis Conference between 1990 and 2014. Figure 33 shows this graph, which has 490 vertices representing the papers and 1;547 edges describing the citations among these papers.

Figure 33 – Graph visualization using the Neighbor-Joining Bundling for the dataset INFOVIS15.



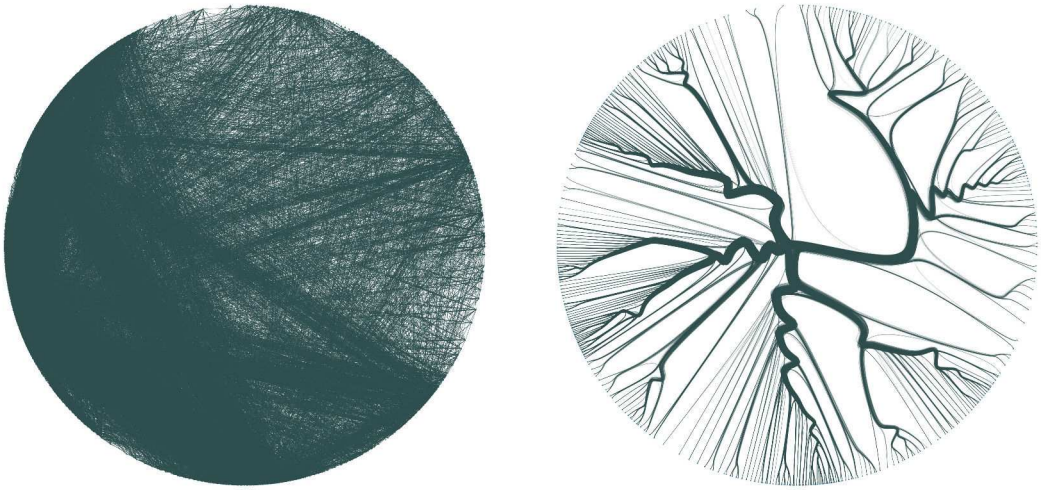
Source: Sikansi and Paulovich (2015) © 2015 IEEE.

This result shows how edges are drawn into bundles. The backbone organizes the edges in paths according to the similarity among documents. The bundling layout enables the detection of larger and smaller bundles. Moreover, it differentiates local bundles from global ones, i.e., bundles generated from edges connecting data objects from the same group from bundles of edges linking data objects from different groups, crossing the center of the visualization. The bundled graph presents different branches sizes, but it does not generate a implicit distortion on the resulting bundles.

However, we noticed problems when applying the neighbor-joining in datasets with similarity distributions different than the commonly presented in text datasets. In such scenarios, the bundling algorithm still reduces the visual clutter, but there is a high variance among the length of each data object branch, which generates distortions on edge paths. In order to show this problem, Figure 34 presents a comparison between the original and the bundled graph of the dataset #NBABALLOT (see section 5.2).

The backbone obtained from this dataset generates a huge contrast between small and large branches. This is noticed by the forms of bundles, not as smooth as the ones presented in Figure 33. Consequently, the neighbor-joining might not be suitable for some datasets, once

Figure 34 – A comparison between the unbundled graph and the one obtained with the Neighbor-Joining Bundling for the dataset #NBABALLOT.



(a) Original graph (b) Bundled graph

Source: Elaborated by the author.

unbalanced backbones generate too much distortion on edge bundles. Moreover, this example reassures the importance of evaluating the backbone because of its ability to express similarity relationships and represent a useful bundling layout.

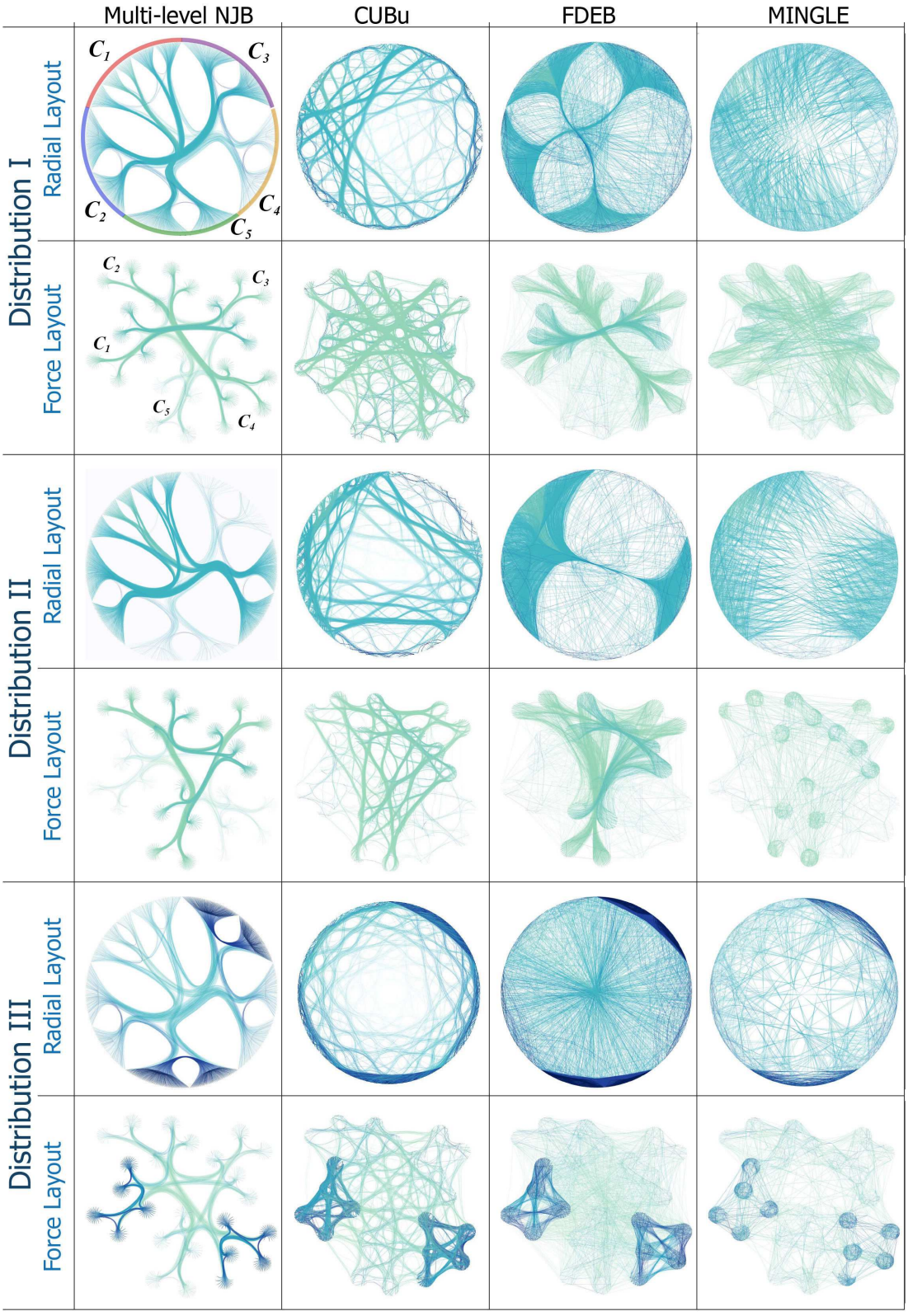
4.4.2 Multi-level Neighbor-Joining Bundling

The Multi-level Neighbor-Joining is a transformation from the previous method to reduce the number of intermediate vertices, to provide a reliable backbone and to decrease the NJ algorithm computational cost. Figure 35 shows the comparison among this method and the selected state-of-the-art techniques. In this comparison, we aim to discuss the readability of bundling layouts and the patterns extracted from each input graph. The Multi-level NJ layouts were generated using two levels and $K = 5$ number of clusters. We use the radial and the force layout to determine the vertices placement into the visual space. This placement is used by all state-of-the-art techniques because they require the vertices position to perform the bundling. As presented in section 4.2, the artificial edges distributions were generated to create edge patterns easily identified when transformed into bundles.

Although the verification of the visual clutter reduction could be a subjective evaluation, we can assert that our edge bundling layout is equal, or better, than some of the most successful edge bundling techniques. Our framework shows the same patterns that are visualized when using other edge bundling methods. Moreover, our method makes the identification of bundles sources and targets easier, while other methods tend to bend edges close to the origin and destination.

Regarding MINGLE and FDEB, the former was not able to create clear patterns that

Figure 35 – A comparison among the results obtained with the Multi-level NJB, CUBu, FDEB and MINGLE using the radial and force layout. In the first image for each layout, the clusters are highlighted following the artificial dataset description.



Source: Elaborated by the author.

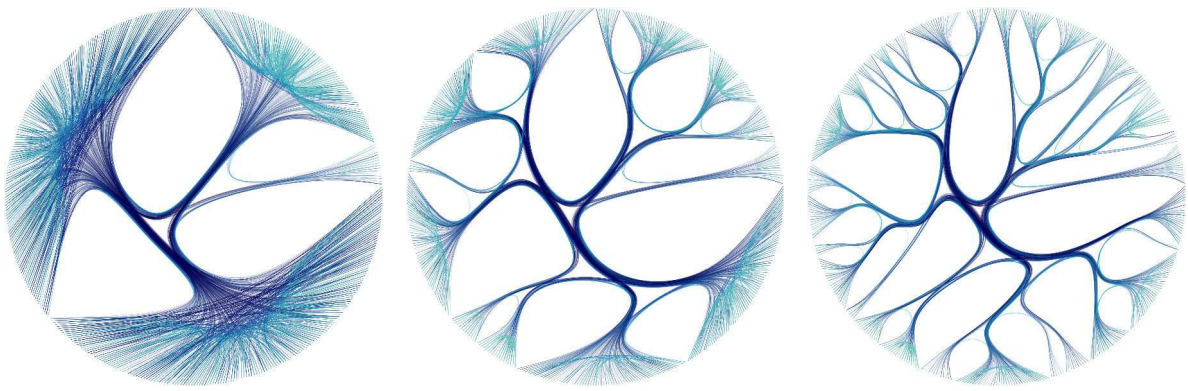
highlight the relationship among groups of edges for any input graph. The latter presented better graphs using the force-directed layout, but did not achieve the same result with the radial one, being the worst case for the DISTRIBUTION II. This distribution represents one group with several connections to two opposite groups in the visualization. However, FDEB failed in recognizing them and merged these two groups into a singleton bundle.

CUBu is the only technique that generates multiple bundles to connect the same pair of groups. This technique layout has straighter bundles and less distortion when compared with other techniques. It is possible to identify the groups patterns as defined by the edges distributions. However, more bundles generate more crossing routes. In particular, the crossings decreased the graph readability, as shown in the graph from the DISTRIBUTION I for the force layout. In this graph, CUBu failed to represent the crossings between two pairs of groups, as presented in the Multi-level and FDEB graphs.

The Multi-level Neighbor-Joining generates more compact bundles connecting different groups. In addition, we can see the expected patterns for all distributions. The technique provided good results for the two methods used to place the vertices. In the radial layout of the DISTRIBUTION III, the bundling result of this technique outstands because the edges connecting elements from the same group are not bent over the virtual circumference where vertices are placed.

Moreover, it is possible to modify the bundling scale through user interaction. As explained in chapter 3, the user can determine the desired number of levels or the minimum cluster size. Figure 36 shows the visualization of one graph with different numbers of levels. For the one-level bundling, the backbone contains only one segmentation of groups. Hence, few and larger bundles are generated, which provides a generic visualization. When the user increases the number of levels, bundles are refined and the relationship among smaller groups are more evident.

Figure 36 – The bundling layout of the graph INFOVIS15 rendered with different number of levels.



(a) One-level bundling (b) Two-levels bundling (c) Three-levels bundling

Source: Elaborated by the author.

The Multi-level Neighbor-Joining Bundling fulfills the aspects listed in our research goals. However, this method requires parameters that make the usage and the evaluation more difficult, such as the number of levels and clusters. While the number of levels affects just the degree of details in the bundling, the number of clusters is a sensitive parameter and demands a prior knowledge of the dataset. Once edge bundling layouts are commonly applied to get a first insight of the data, it is contradictory to require a previous knowledge about the dataset. To address such limitation we devised the STree backbone, next evaluated.

4.4.3 Similarity-driven Edge Bundling

We evaluated the Similarity-driven Edge Bundling (SDEB) with a qualitative analysis of the bundling results using the radial and the H-Tree layout for vertices placement. Once more, we replicated the same vertices placement given by our backbone in the other techniques. Starting with the radial layout, Figure 37 shows the comparison using the three datasets with two edges distributions, as presented in section 4.2. For all techniques, we determined the best possible parameters for such input graphs. For the SDEB, we set a filter of intermediate vertices to drop the two first levels of the Similarity Tree (more details about this filter will be presented in section 4.6).

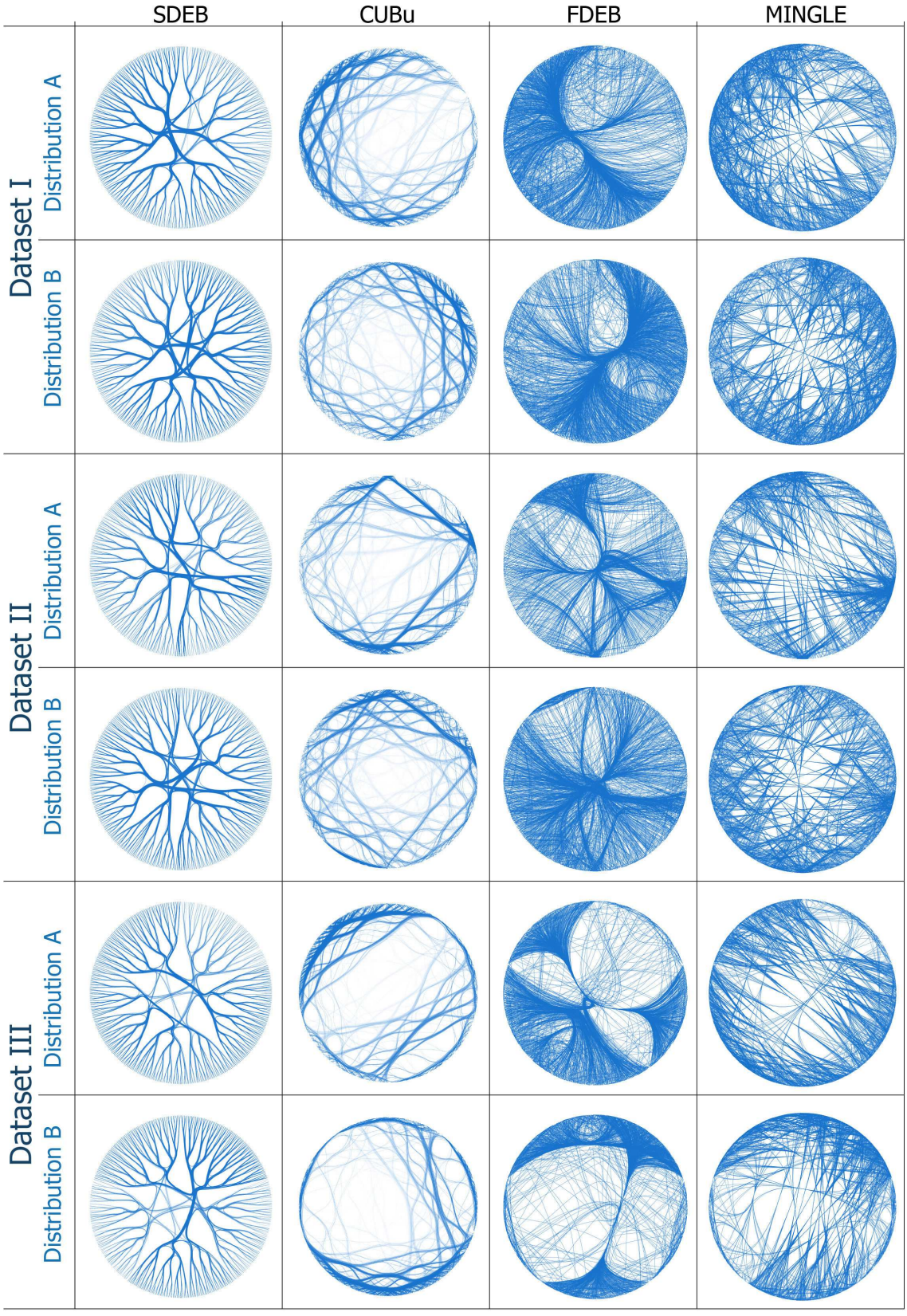
Considering that the backbone construction divides the data objects in two groups at each interaction, this experiment used different datasets, with 3, 4 and 5 clusters. Therefore, we avoid the usage of a biased input graph with only two clusters. The result shows that our layout is not affected by these different numbers of clusters. In all cases, the backbone remains balanced and the clusters are still recognizable.

For all input graphs, we noticed that FDEB and MINGLE failed to present well defined bundles. The worst results were presented by FDEB, which could not create any bundle and resulted in a completely disorganized layout. According to its behavior, MINGLE created bundles with a few edges. Even though it has some bundles, the layout is congested in the border of the visualization. Moreover, FDEB and MINGLE presented several "outlier" edges, i.e., edges that are not grouped into any bundle. Regarding CUBu results, we noticed a less cluttered layout. The expected dense connections between groups are recognizable, as shown in the adjacency matrix presented in Table 1. However, this technique generates bundles using lesser bent lines, leaving the center of the visualization empty and the board loaded with too much information.

Different than CUBu, SDEB does not agglomerate the edges around the border of the visualization. Instead, our technique follows the backbone, which distributes the bundles through all visual space. The center of the visualization is the most important area, where several bundles are divided and there are crossing routes. Most crossings in the SDEB layout happen in different directions, so there is no misinterpretations of the crossings as it may happen in the CUBu layout.

Figure 38 shows in detail the differences between CUBu and SDEB. CUBu bundled all

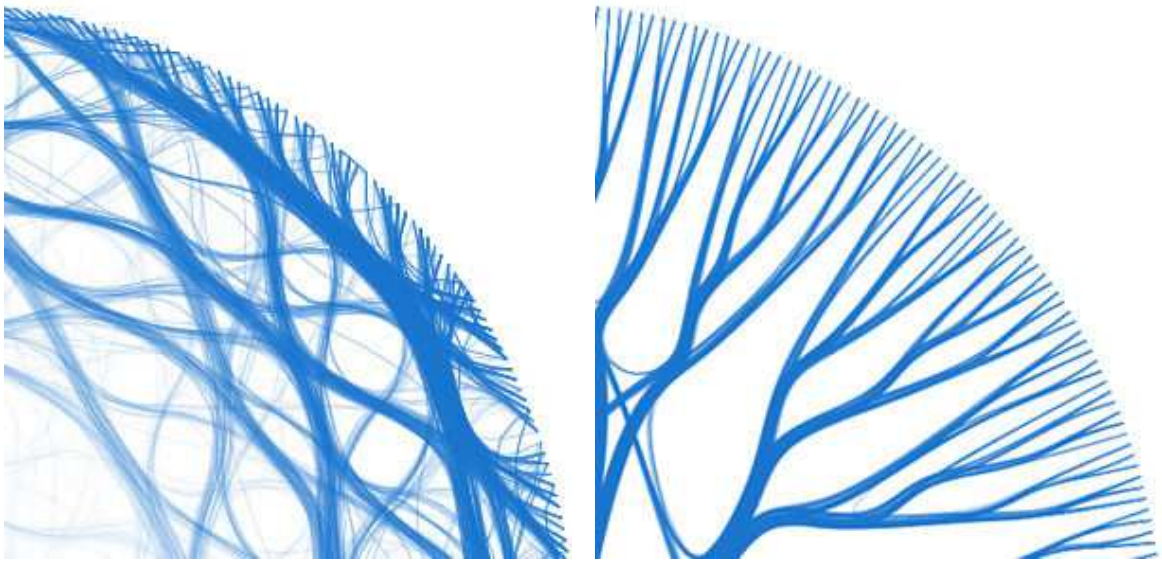
Figure 37 – A comparison among the results attained by the SDEB, CUBu, FDEB and MINGLE using the radial layout.



Source: Elaborated by the author.

edges close to their source/target vertices, generating an ambiguity concerning the edges flow. On the other hand, SDEB drawn edges following a direction which is perpendicular to the tangent on their source/target. The agglomeration of edges happens at different levels and not all edges at the same time. In addition, CUBu seems to mix several bundles in a way that disregards each bundle meaning. The same does not happen with the SDEB, in which it is possible to identify the group of vertices represented in each bundling.

Figure 38 – A zoom in on part of the visualization produced by the techniques CUBu and SDEB.



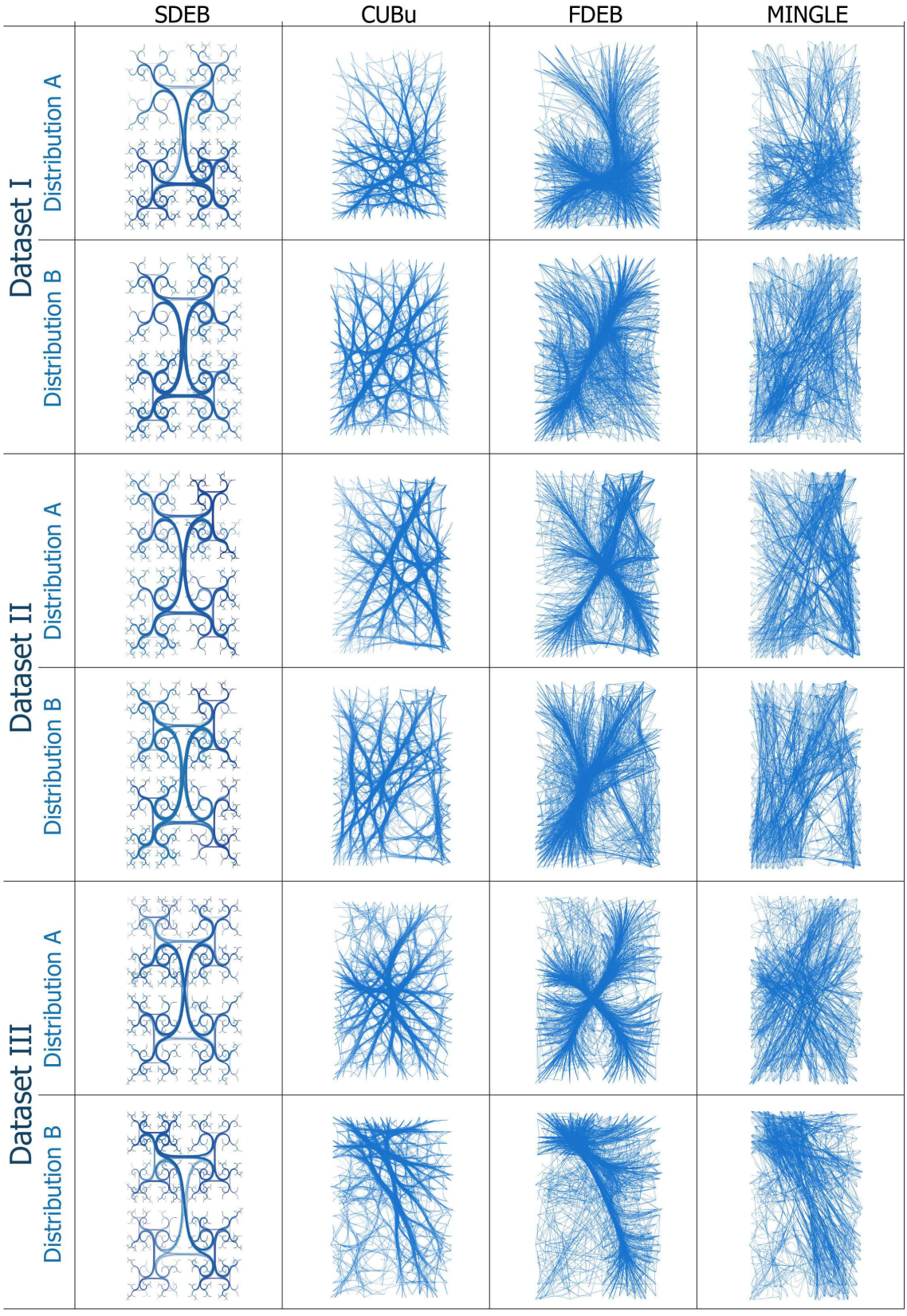
(a) CUBu (b) SDEB

Source: Elaborated by the author.

Figure 39 shows the same input graphs from Figure 37 but the backbone is positioned using the H-Tree layout. As explained in chapter 3, the H-tree layout focuses on the vertices, spreading them through the visual space. Even tough the techniques share the same vertices position, there is a huge contrast, not noticeable on the radial layout, between the SDEB and other techniques. This contrast happens because the other techniques generate the bundles only considering the spatially-similar edges trajectories, while our technique follows the H-Tree layout. Once more, we did not apply any enhancements developed for our technique (these are discussed in the next section).

This comparison shows that all state-of-the-art techniques were able to produce an edge bundling layout that highlights the major pattern of each graph. FDEB and MINGLE presented competitive results, showing a better performance when the vertices are spread over the visual space. However, their graphs do not reach the same clutter-reduction observed in CUBu and SDEB. Techniques MINGLE and FDEB created distinct flows that agglomerate almost all edges for each direction observed in the input graphs, while CUBu created separated bundles in several directions.

Figure 39 – A comparison among the results attained by the SDEB, CUBu, FDEB and MINGLE using the H-Tree layout.



Source: Elaborated by the author.

Our technique takes advantage of the H-Tree layout to produce more compact bundles and a less cluttered layout than any other technique. In addition, vertices placed in the center of the visualization are easier to be identified. Although there is some overlapping of edges over vertices, this occurs less often than in the other techniques, in which centralized vertices are completely hidden by the bundles. However, bending edges according to the H-Tree layout forced the bundles to repeat the pattern existent in the fractal structure, which makes the edges patterns a little harder to be identified.

Despite that, we can recognize the edges patterns by analysing the bundles density. Moreover, the compacted view provided by the SDEB using the H-Tree layout improves the identification of sectional shapes. This is evident in the graph from the DATASET II DISTRIBUTION A, where there is a local concentration of strong bundles in the group positioned in the right-bottom area. In other techniques, this local behavior is obfuscated by the largest (global) pattern.

4.5 Computational Cost

Being $|V|$ and $|E|$, respectively, the number of vertices and edges of a given graph, we outline the computational cost of each stage of our bundling framework as follow:

- First, to compute the Similarity Tree, the algorithm has time complexity $O(|V|\log|V|)$;
- Second, to compute the edges path, the framework takes two steps: it computes the path between the root and each leaf, which has time complexity $O(|V|)$, and then compute the path of each edge, which has time complexity $O(|E|)$. Therefore, the total computation cost of this process is $(O(|V| + |E|))$;
- Third, to perform the backbone placement, the circular and the H-Tree layout have time complexity $O(|V|)$.
- Finally, to draw each edge though its control points, computing the edge transformed by the parameters, the computational cost is $O(|E|)$.

Thereby, the Similarity-driven Edge Bundling has computational cost $O(|V|\log|V|)$ defined by the similarity tree construction. Just for comparison, the the Neighbor-Joining has time complexity $O(|V|^3)$ and the UPGMA has time complexity $O(|V|^2)$. Both methods are more expensive than the Similarity Tree, apart from the layout issues commented before.

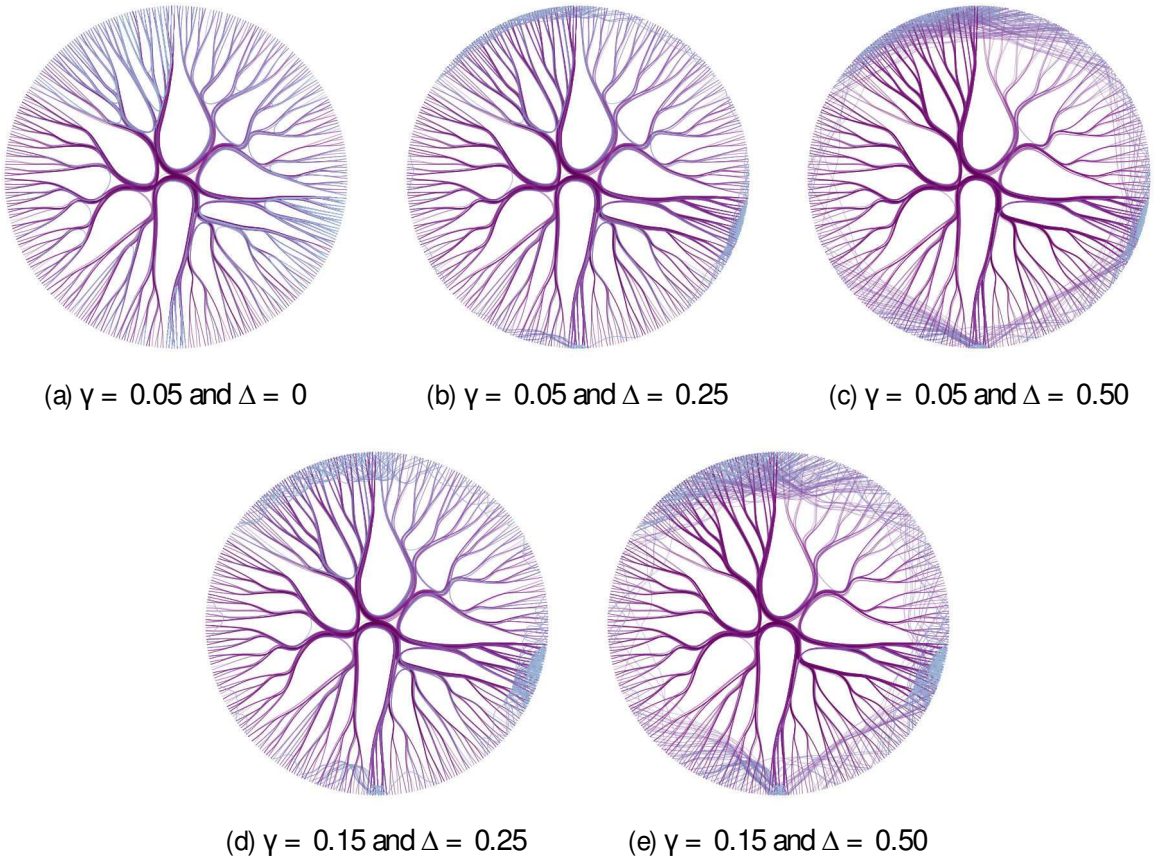
4.6 Bundling Enhancements

In this section, we report the results of three enhancements designed to give the user a set of options that transforms the bundling visualization. These transformations are the Adaptive-β,

the intermediate vertices filtration and the multiscale visualization.

The Adaptive-β enables an adaptive control of tension of each edge. In our method, the tension, i.e., the strength with which edges are attracted into the backbone structure, is defined by three parameters: β, γ, and Δ. As established by the Equation 3.4.2, the tension is defined by an activation function, where β is the tension for activated edges and γ is the tension for inactivated ones. Δ defines the minimum size for activation. Figure 40 shows different examples of those parameters for the same input graph.

Figure 40 – Multiples bundling configurations in the radial layout by varying the parameters γ and Δ



(a) $\gamma = 0.05$ and $\Delta = 0$ (b) $\gamma = 0.05$ and $\Delta = 0.25$ (c) $\gamma = 0.05$ and $\Delta = 0.50$

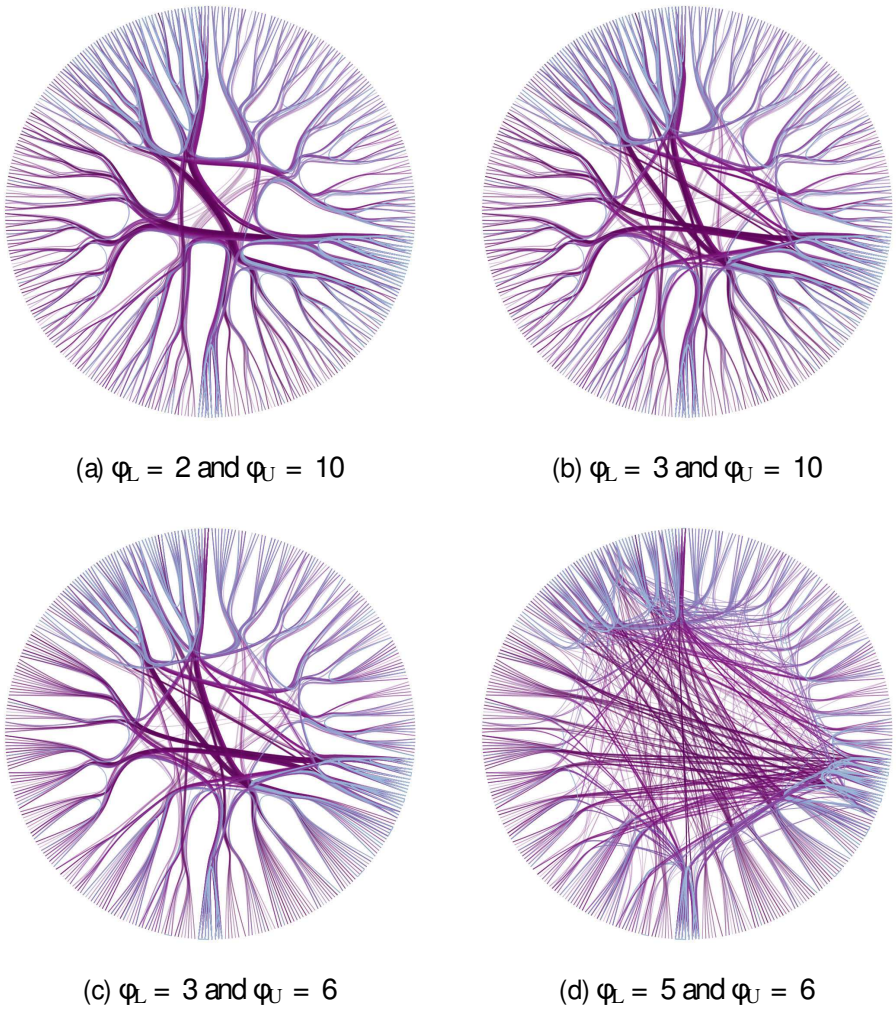
(d) $\gamma = 0.15$ and $\Delta = 0.25$ (e) $\gamma = 0.15$ and $\Delta = 0.50$

Source: Elaborated by the author.

From these results, we see how the parameters γ and Δ can be changed to improve the visualization, revealing hidden local information. When we set $\Delta = 0$ (Figure 40a), short edges are hardly identified because they are mixed with long ones, which are most distinguishable. When we increase the value of Δ (Figure 40b and Figure 40c), it separates short edges from bundles, showing concealed information, such as the groups of similar vertices with more connections. These connections are highlighted when we vary γ because they are pushed towards the center of the visualization (Figure 40d). However, the unbundled edges generate a noise that must be controlled (Figure 40e) and high values of γ and Δ might make the visualization worse.

The second enhancement is the intermediate vertices filtration. In this modification, the user sets the interval $[\varphi_L; \varphi_U]$ of levels that should be considered when the edges are bent over the backbone. Therefore, the edges are drawn only using the intermediate vertices in that interval. Figure 41 shows four variations of the intermediate vertices filtration. These examples show the effects of dropping intermediate vertices from the root and from the leaves of the backbone. Removing intermediate vertices closer to the root creates direct connections between groups in deeper levels, resulting in a more detailed visualization. In the other side, dropping intermediate vertices closer to the leaves generates bigger and more generic groups.

Figure 41 – Multiples bundling configurations in the radial layout by varying the interval of intermediate vertices used to bend the edges.



(a) $\varphi_L = 2$ and $\varphi_U = 10$ (b) $\varphi_L = 3$ and $\varphi_U = 10$

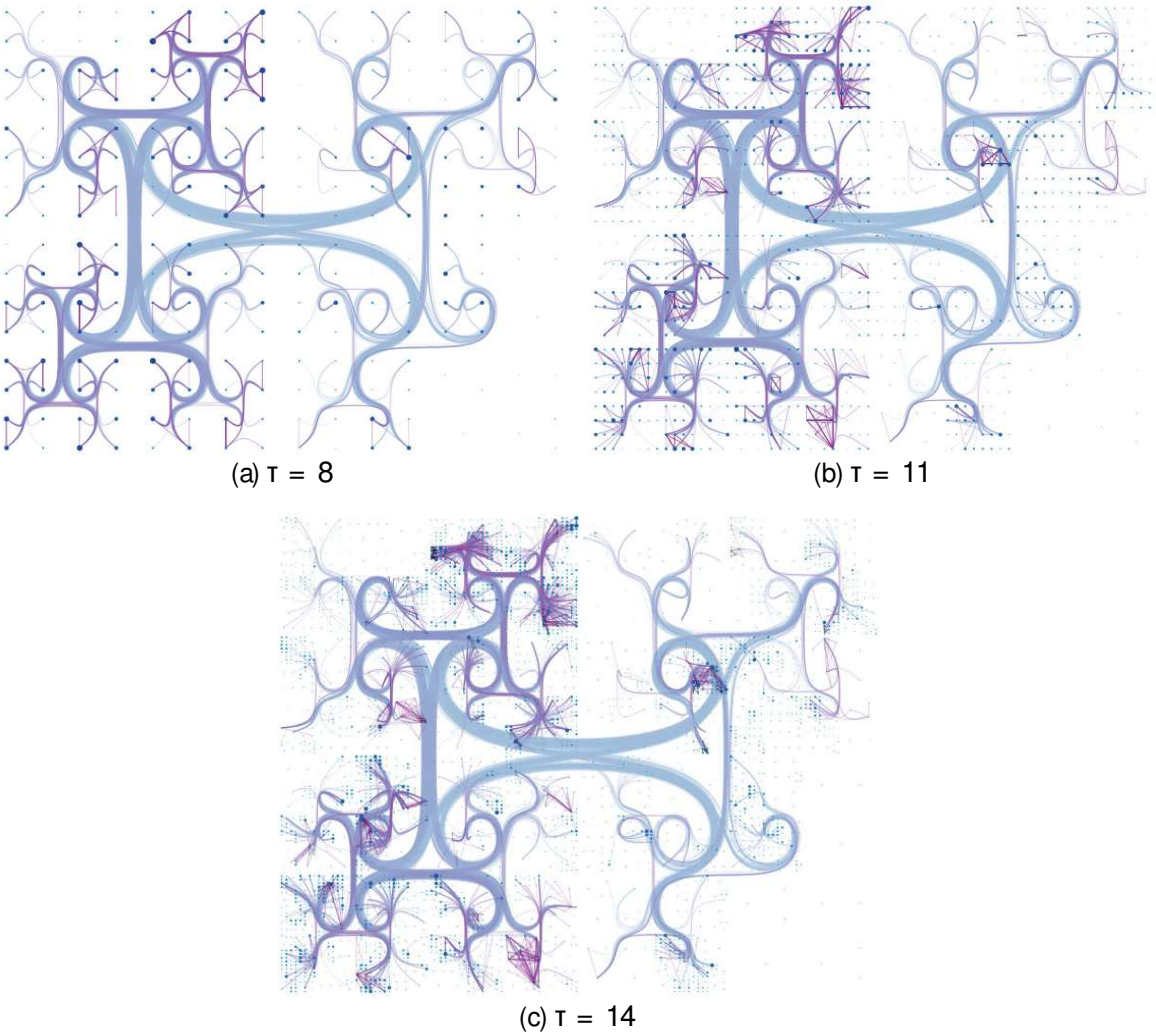
(c) $\varphi_L = 3$ and $\varphi_U = 6$ (d) $\varphi_L = 5$ and $\varphi_U = 6$

Source: Elaborated by the author.

Finally, the edge bundling layout can be modified by the multiscale visualization. This feature enables the user to determine if the backbone placement will draw the entire backbone or the branches will be cut in a maximum level (τ). This interaction summarizes the branches and might be used to reduce the size of large datasets, since the user can control the amount of vertices placed in the visualization. Figure 42 shows three different values for the global threshold. When

the maximum depth is small, it generates a coarse visualization with fewer vertices representing the set of their children (Figure 42a). When the maximum depth is increased, the visualization shows more detailed information and contains more edges in local areas (Figure 42b). For larger values, local information is more readable and there are final vertices in less dense areas, as well as other ones that still represent its children (Figure 42c).

Figure 42 – Multiples bundling configurations using the H-Tree layout by varying the maximum depth in which the vertices are positioned.



(a) $\tau = 8$ (b) $\tau = 11$

(c) $\tau = 14$

Source: Elaborated by the author.

4.7 Final Remarks

In this chapter, we presented the evaluation of our technique and its comparison with the state-of-the-art. We divided our evaluation framework into two main experiments. The first one showed the efficiency of the Similarity Tree to represent similarities among the original vertices. Two metrics demonstrated that our technique is competitive with classic similarity

based three constructions NJ e UPGMA. Moreover, our algorithm has a lower computational cost and produces a more balanced tree, which is better for bundling purposes.

In the second stage, we evaluated the edge bundling layout guided by the Similarity Tree. We compared our technique with three of the most successful state-of-the-art techniques, each one from a different group of techniques presented in chapter 2. We attested that our technique outstands others by producing more meaningful bundles using both radial and H-tree layouts. The experiments were made using artificial data and datasets commonly used to assess similarity-based methods, such as trees and multidimensional projections. Since we had prior knowledge about the data, we could verify the expected behaviors of the bundling layout. Finally, we presented enhancements to the SDEB layout that improves the edge bundling readability.

In the next chapter, we present a set of applications with real-world datasets, where we can see the advantages of the edge bundling layout to visualize data with complex similarities relationships and without any prior knowledge.

CHAPTER

5

APPLICATIONS

5.1 Initial Remarks

In this chapter, we show the usage of the Similarity-driven Edge Bundling with real-world datasets. In these scenarios, the similarity relationships and data distributions are not well-defined as they were in the artificial datasets. Hence, we do not have prior knowledge to verify the bundling meaning. These applications are examples where our technique produced a good edge bundling visualization and allowed the analysis and detection of patterns. We present three applications, each one with a different dataset. The following section describes the datasets employed in these applications.

5.2 Datasets

The applications use datasets found in related works or commonly used in others fields of information visualization. In addition, we also collected data during this research. In this section, we present each dataset.

The InfoVis 2004 Contest - The History of InfoVis (INFOVIS04) (FEKETE; GRINSTEIN; PLAISANT, 2004) and the Visualization Publication Dataset (INFOVIS15) (ISENBERG et al., 2015) are examples of citations network datasets. The first one has 433 vertices and 1;446 edges, while the second one has 384 vertices and 1;633 edges. Citations networks are a well-known subject in graph visualization. We extract the similarity among papers from their abstract and other meta-data available.

The #NBABALLOT was built during this research and consists of a collection of posts extracted from Twitter (commonly called tweets). We created this dataset to apply our technique in time-varying information. We selected tweets published with the hashtag #NBABALLOT between December 19th, 2014 and January 21th, 2015. The users used this term to vote in a

NBA[1] player to compose one of the two teams of the 2015 NBA All-Star Game. This election selected 5 players for each team, the Eastern All-Stars and the Western All-Stars. During this stage, more than 2 millions of posts were collected.

After collecting the tweets, we processed the raw data to create our dataset. The data used to represent each player and calculate the similarity among them was extracted from the official league statistics page[2], which covered the major statistics measures of basketball matches during that season. Each edge connects players that had received votes from the same user. This graph was segmented by days of voting, which we called a voting frame. The final graph consists of a dynamic graph with 436 vertices, 11;793 distinct edges and 34 frames. Considering the whole list of pairs, we found 565;829 associations between two players that received a vote from some user in the same day.

Finally, to present the application of our technique in a large dataset, we use the AMAZON GROCERIES REVIEWS from SNAP[3] (MCAULEY; LESKOVEC, 2013). This dataset represents more than 500;000 products from the Amazon website[4] and their respective information, such as category, similar products and a set of reviews written by clients. We filtered this data to create a graph with only groceries products. The resulting graph has 8;700 vertices and 129;407 edges. Each edge represents a pair of co-purchased items. The similarity among products was calculated using the reviews published by customers.

5.3 Visualization of citations networks

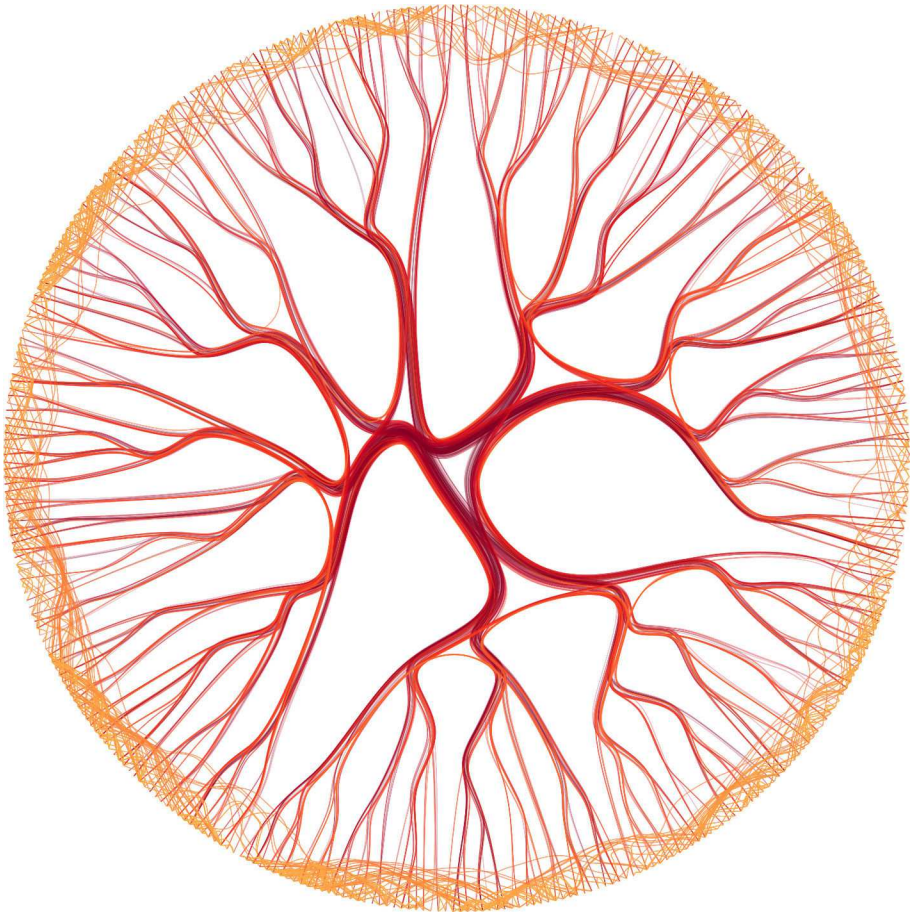
Our first application concerns the exploration of citations among scientific papers. Citations networks are often applied in graph visualization to detect patterns, such as citations among researchers, research areas or publication venues. A graph of citations from the dataset INFOVIS04 was presented by Ersoy et al. (2011) using the SBEB technique. This technique requires the original placement for each vertex, and Ersoy et al. (2011) used multidimensional scaling with the least-square projection (PAULOVICH et al., 2008). Differently, our technique uses the vertices placement obtained from the backbone to build the visualization.

In our application, we use the recent INFOVIS15, selecting a subset with papers that have at least 5 citations. This subset has 384 vertices and 1;633 edges. First, we extracted a term vector using the available data (title, authors and abstract) of each paper. Then, the distances among papers are calculated using the cosine-based (SALTON, 1991) dissimilarity from the term vectors of the papers. The similarity between two documents is given by the inverse of their distances. Figure 43 shows the visualization using the radial layout. We set the parameters

[1] National Basketball League, the famous North American basketball championship
[2] http://stats.nba.com/
[3] https://snap.stanford.edu/
[4] https://www.amazon.com/

$\beta = 0.97$, $\gamma = 0.1$, $\Delta = 0.12$, and the filtering function of control points was set with parameters $\varphi_L = 2$ and $\varphi_U = 7$. Edges' colors represent their sizes.

Figure 43 – Edge bundling layout from the dataset INFOVIS15.

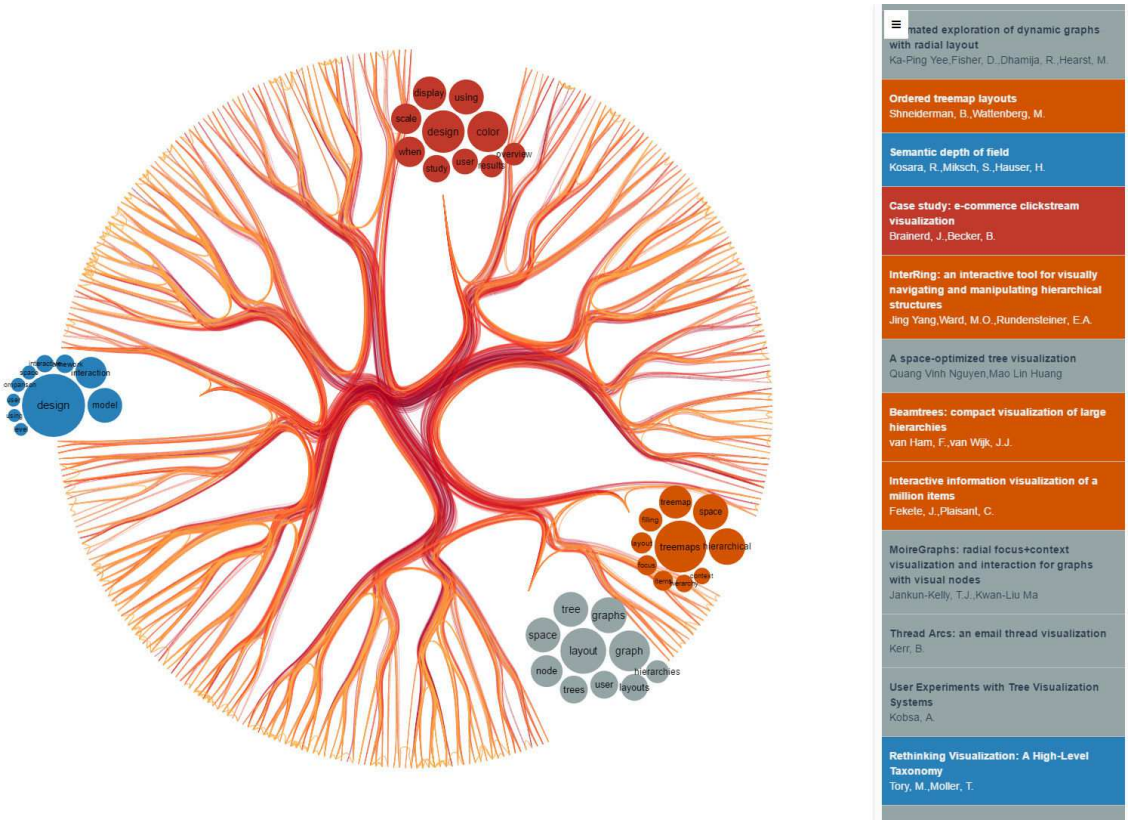


Source: Elaborated by the author.

Our first observation concerns the non-bundled edges. Because close vertices represent similar papers that probably share the main subject, we could expect a large number of short distance edges. However, the distribution is not regular, which may indicate that some topics concentrate more citations among related papers, while other groups cite similar papers less frequently.

We also observe relationships among different groups, although it is hard to make conclusions by only looking at the complete graph. Taking advantage of the hierarchical structure, the user can interact with the visualization to obtain a more detailed visual representation. One possible interaction is the graph summarization. By applying the multi-level control in certain branches, the user can select groups of similar vertices and visualize them as a single vertex. In that visualization, the selection hides the grouped edges and shows data information about the group. We replace the hidden vertices by a tag cloud with the most frequent terms. Figure 44 shows the summarization of four groups, each one identified by a different color, the user also sees a list with the selected papers.

Figure 44 – Edge bundling layout summarization by collapsing branches from the backbone. A tag cloud describes the most frequent terms found in the papers showed in the auxiliary list on the right. The color of each entry identifies the group in which the paper is positioned in the graph.



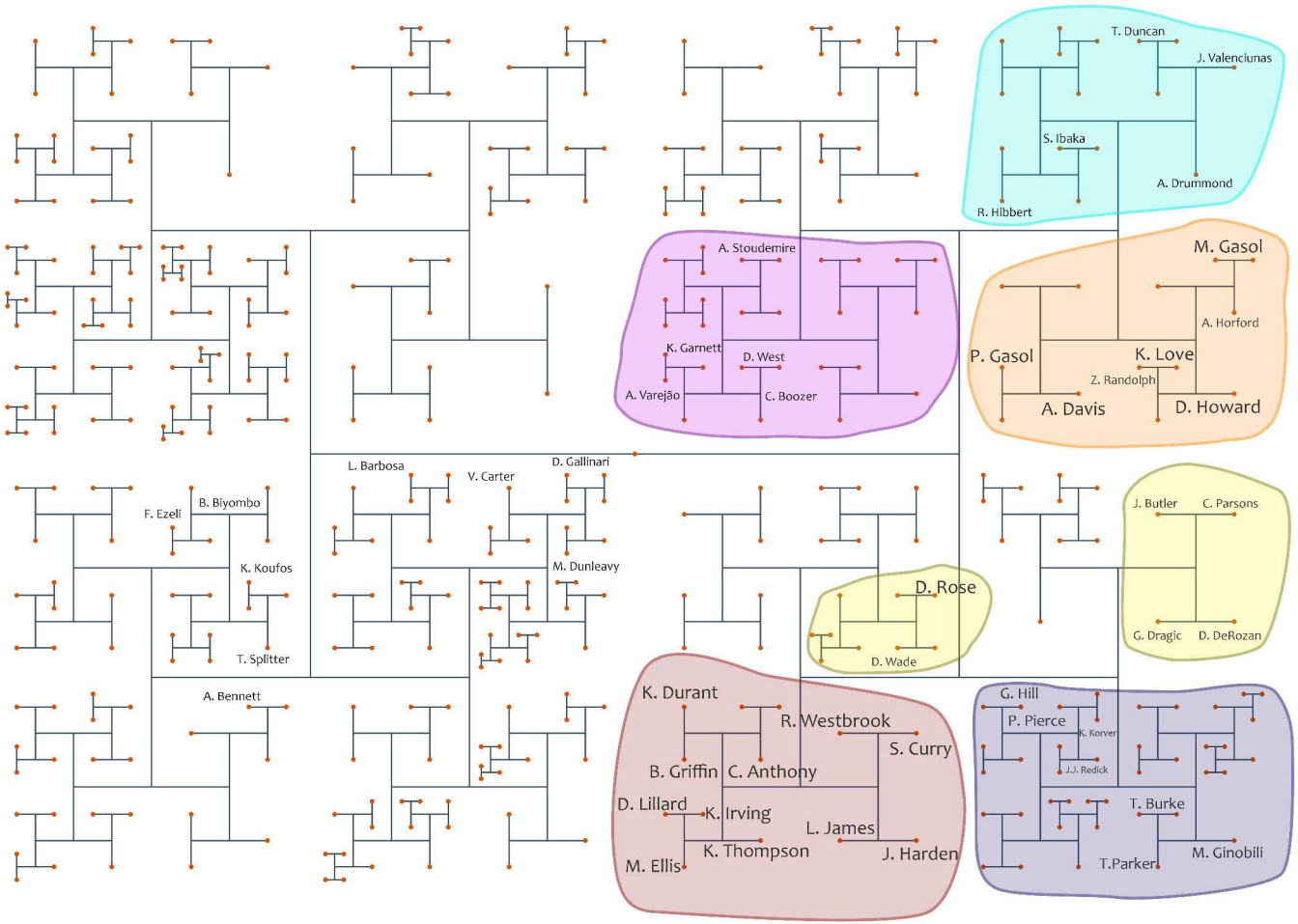
Source: Elaborated by the author.

5.4 Visualization of social networks data

An important aspect of social networks is the fact that data changes over time and it may be related to how people react about different events. This kind of data includes not only personal relationships, but also content based information, such as texts, videos and images. In our case, we want to visualize how users from Twitter interact with the election of the players for the 2015 NBA All-Star Game, explained in section 5.2. This visualization shows how the Similarity-driven Edge Bundling can be used to visualize a dynamic set of edges, when the vertices and the similarity relationship among them are static.

We start this discussion analysing the backbone construction. In this graph, each vertex represents a candidate for the 2015 NBA All-Star Game. The similarity among players is determined by the statistics commonly measured in a basketball game (e.g., points, rebounds, assists, turnovers, field goal percentage, three point percentage, blocks, etc). Figure 45 shows the backbone obtained using the statistics collected in the same season, before the election started. Moreover, some groups are highlighted to help identifying important players, which may be positioned close to each other due to their similar attributes.

Figure 45 – Backbone of the dataset #NBABALLOT. Some vertices are labeled to show some players in the backbone. Colored groups display noticeable players that share characteristics and attributes.



Source: Elaborated by the author.

We perform an analytical evaluation of the dataset considering the common knowledge of the field. Part of our analysis uses the terms that define the position of players in a basketball game. Basketball players are commonly assigned to one of five positions in the court, which are Point-guard, Small-guard, Small-forward, Power-forward and Center. The first two positions are usually called guards, while the last two are usually called post. Small-forward players are versatile players that may be associated to both post and guard positions, according to the team strategy.

The first noticeable event is the first division in the backbone construction. It splits the players into two groups. The first group represents the more active players (i.e., starters or engaged bench players), while the second group is formed by less active players (i.e., players that usually play for a few minutes). The right side of the backbone concentrates the most famous players of the league, which are on the top of many statistics of the season. On the contrary, players from the left side are less relevant players.

After the second division, it becomes harder to analyse the left side, because players with a few minutes per game have inaccurate statistics. Regarding the right side, the backbone clearly separates post players, placed at the top, and guard players, placed at the bottom. This is an evidence that we can find similar players, according to the position they act on the court, only using their statistics.

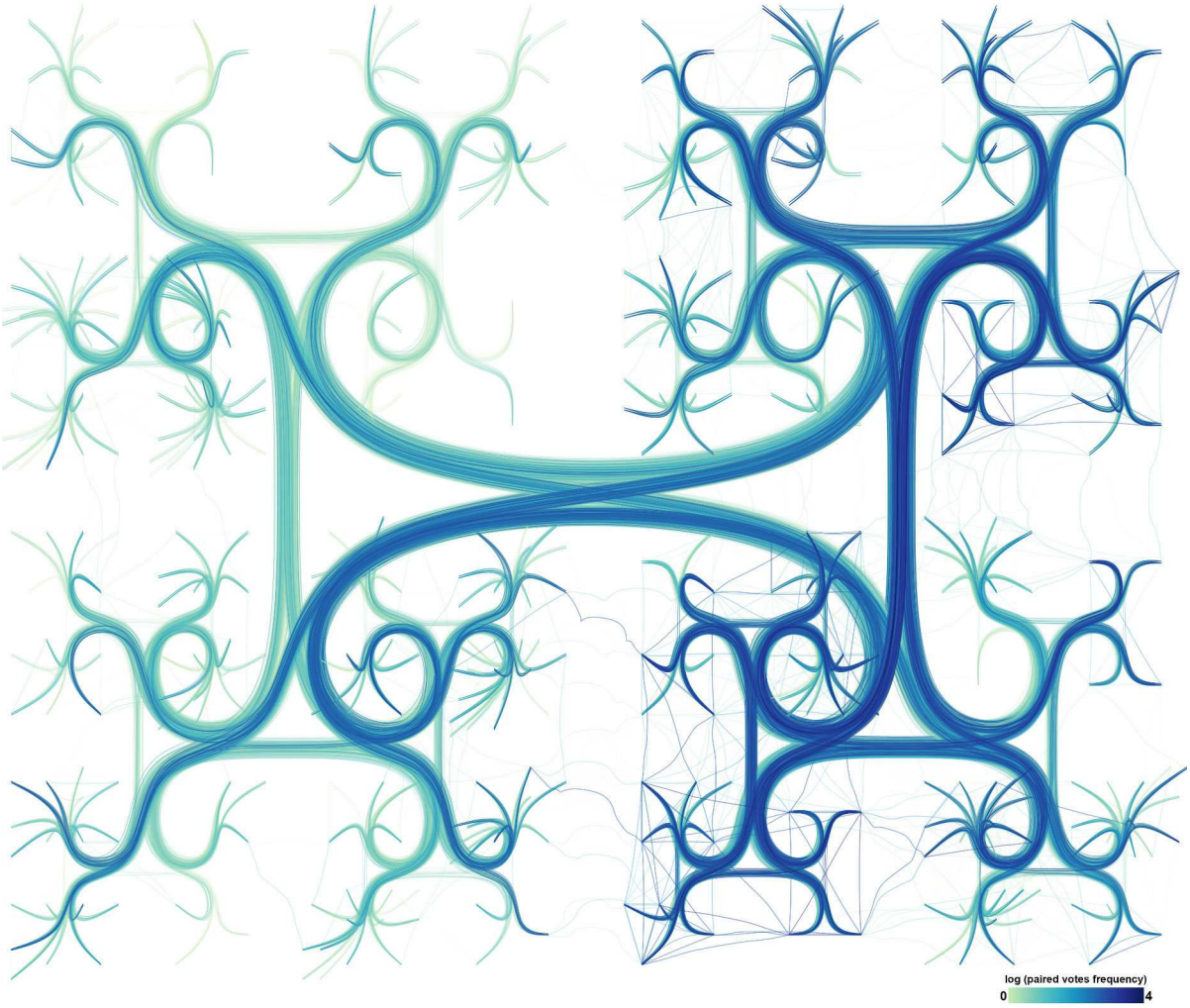
The backbone shows more similar players in deeper levels. The red group encompasses the ones considered to be the best players in the league. These players are known to have a good performance in most statistics measures, which makes them the most important player of each respective team. The blue group identifies players with similar abilities, but they do not have a leading role in their teams. Groups identified in yellow are similar to the red and blue groups. The players from the yellow groups are important ones, but they present a good performance in only a few metrics.

On the top, the labeled post players are divided in three different groups. The orange and cyan groups embrace the best post players in the league, although the cyan one covers players with less impact in their team statistics. The purple group depicts good players with a supporting role in their teams. It is noticeable that there are less post players than guard ones in the league. This is because teams usually play with only two, or even one post player, while there are three to four guards.

Figure 46 shows the edge bundling layout of the entire dataset (i.e., not divided by frames). When analyzing the graph, the first recognizable aspect is the pattern of connections between the red and the orange groups identified in Figure 45. These edges show a voting pattern, which consists of users voting for guard and post players. Therefore, the users split their votes in the two categories, even though this is not a requirement of the election. Moreover, short distance edges, which connect similar players, are easily identified. There is a large amount of unbundled edges connecting elements inside the red, yellow and blue groups, while there

are fewer occurrences of those in other areas. The visualization also shows that users usually vote in more players from those groups, while they select fewer post players. This behavior was confirmed in the election's result, in which just three players (Anthony Davis, Pau Gasol and Marc Gasol) out of ten were elected from the top-right side.

Figure 46 – SDEB layout of the dataset #NBABALLOT with parameters $\beta = 0.96$, $\gamma = 0.05$ and $\Delta = 0.15$

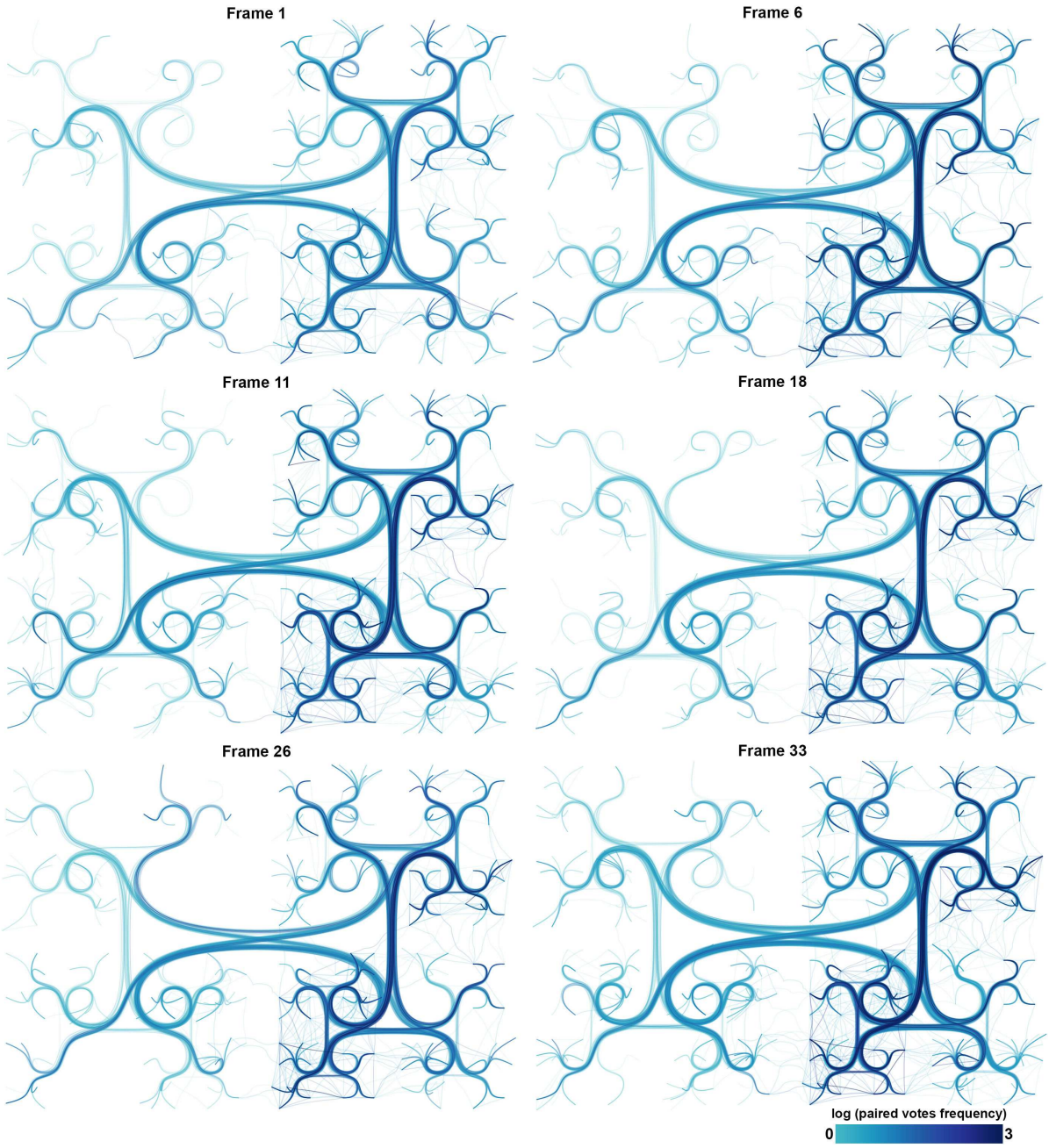


Source: Elaborated by the author.

Furthermore, we can analyse different voting frames separated to understand how the votes behavior changed over the time. Regarding the visualization of time-varying information, there are two main metaphors to visualize such kind of data: animation and small multiples (BOYANDIN; BERTINI; LALANNE, 2012). The latter is preferred by some studies since it can show different frames together (ARCHAMBAULT; PURCHASE; PINAUD, 2011). Considering our framework, both methods can be used because the backbone is the only information used to bend edges and it guarantee the context. Once this application is composed by a sequence of static frames, the small multiples metaphor can be easily applied. Figure 47 shows the #NBABALLOT graph of votes for six selected frames using small multiples. All graphs were generated with parameters $\beta = 0.96$, $\gamma = 0.05$ and $\Delta = 0.2$. The edges are colored according to the number of

occurrences of paired votes.

Figure 47 – Small-multiples visualization for multiple voting frames from the dataset #NBABALLOT



Source: Elaborated by the author.

Comparing the different frames, the same major pattern identified in Figure 46 is also visible. However, we can observe slight modifications in the density of each bundle, such as in frame 1 and 11. Moreover, the group of less important players (left-side) concentrates fewer edges, which provides interesting insights. There are some outlier players that have several connections with the opposite side in some frames. For example, there are edges from the top-left group that only appear in frames 1, 6 and 33. A further investigation can map this behavior with events related to the election, such as game days or marketing campaign to ask for votes.

The main difference between our method and previous techniques used in the context of dynamic graphs concerns the stability. Techniques that compute bundles based on edges would create different bundles based on edges spatially distribution. Therefore, the same edge may be placed in different bundles when comparing different frames. This shows the poor stability of those techniques, resulting in potential context loss and misinterpretation. Our technique draws edges in the same bundle disregarding changes over time. Thereby, opacity and colors show the amount of edges in each bundle. Although this may be harder to analyze, it provides a more faithful information.

5.5 Similarity Bundling on large datasets

Most bundling techniques have reported results with datasets restricted to few hundreds of vertices and thousands of edges. For example, the United States Airlines has, with some variations, 1;790 vertices and up to 9;798 edges. Those are larger values for a common graph drawing algorithms, thus subject to the visual clutter. However, those techniques do not tackle larger graphs with, for example, more than 10;000 vertices or 100;000 edges. The techniques that handle such size of input, like MINGLE, KDEEB and CUBu, claim it only because they are fast enough to calculate bundles, not because of their layout scalability.

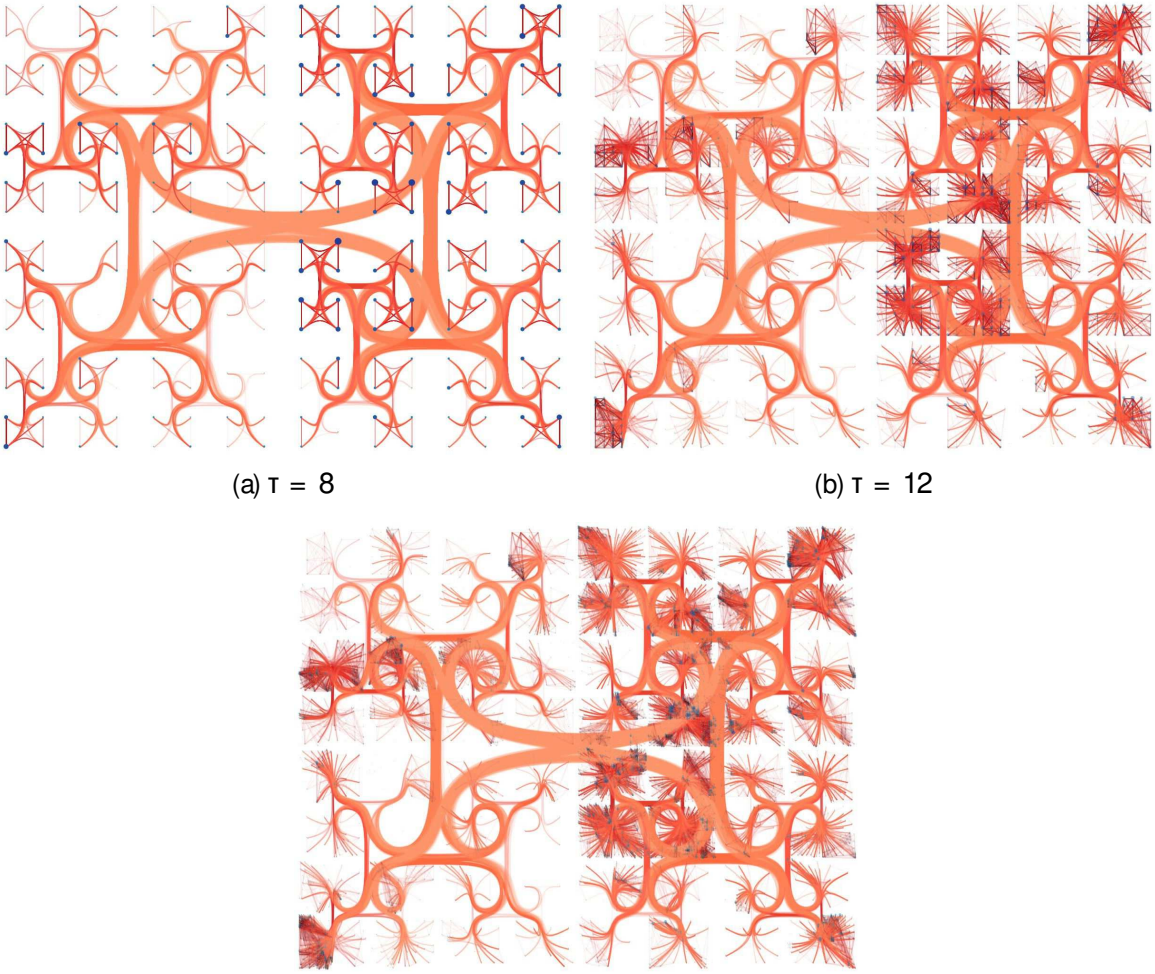
Usually, the number of edges does not affect the bundling performance. Because overlap edges is the bundling goal, more edges just imply in more overlapping. However, the number of vertices is a critical aspect, once the user needs to be able to detect and identify different vertices. Moreover, different from other visualization metaphors, such as scatter-plots or projections, elements overlapping are a major concern in graph visualization. In fact, the vertices must be visible and well represented through the visual space. Our strategy to avoid vertices overlapping is the multiscale bundling.

In this section, we discuss how the multiscale bundling layout can provide useful insights on large graphs. We chose the Amazon Groceries Reviews from SNAP[5] (MCAULEY; LESKOVEC, 2013). This graph has 8;700 vertices, representing each product, and 129;407 edges, representing a pair of products purchased together. To avoid the representation of this amount of vertices, we used multiple values of the multiscale threshold τ. Products under the level of τ will be represented by their clusters. Edges connecting elements inside the same cluster disappear, because their source and target are being represented by the same vertex in the visualization. The user can manipulate this value to get different levels of details. Figure 48 shows three layouts, varying the maximum depth in which the backbone is rendered. The similarity among products was calculated using the bag-of-words vector of multiple products reviews.

The different layouts show how the level of detail increases when deeper levels of the backbone are rendered, thus reducing the amount of filtration. When $\tau = 8$ (Figure 48a), only

[5] https://snap.stanford.edu/

Figure 48 – Comparing multiple scales for the bundling layout of the dataset Amazon Groceries Reviews .



(a) $\tau = 8$ (b) $\tau = 12$

(c) $\tau = 16$

Source: Elaborated by the author.

248 vertices are visible, with almost all vertices are representing clusters of products. If the maximum depth is increased, the layout shows more vertices. The layout with $\tau = 12$ shows 2;108 vertices (Figure 48b) and the one with $\tau = 16$ shows 5;856 (Figure 48c), which means that only few vertices still represent clusters of products. In such cases, we also observe the congestion of edges inside more populated groups.

To extend our analysis, we fixed $\tau = 10$. Figure 49 shows the bundling layout of this graph. In addition, we highlighted the 64 groups of products with the most common topics found in their reviews. By labeling the groups, we can see which products belong to each group. Some neighbor groups share the same topics, indicating that these categories were defined in a higher level and refined in the lower ones. For example, all groups in the bottom-left branch are labeled with "Tea". Another noticed behavior is that the first division segments liquid and solid groceries. The bundling layout was created using the parameters $\beta = 0.95$, $\gamma = 0.1$ and $\Delta = 0.13$. This configuration also highlight the dense groups, with more connections among their elements and

their neighbors.

Looking for more detailed information about relationships among groups, the user can select multiple groups and filter edges. Figure 50 shows the edge filtration from three different groups: the bottom-left "Tea", "Cherries" and "Organic and Baby Food". In this visualization, we set $\beta = 0.90$ and $\Delta = 0$, because we were not focused in short edges. Although the reduced value of β increases the edges overlap over vertices, it improves the observation of multiple groups, which were the main goal of this example. This is necessary because edges opacity are not effective in a graph with such amount of edges, and, consequently, too much overlapping. Moreover, the selection highlights how groups are related with other ones. A useful insight observed is the difference between how the groups black and green are related with their respective opposite products. While a wide bundle connects edges in black directed to this opposite group, a narrow green one represent the inverse direction.

Our analysis shows that the Similarity-driven Edge Bundling is able to present several insights about the relationship among co-purchased products, such as categories of products that are usually purchased together. Furthermore, the clear layout produced by our technique facilitates the detection of vertices and clusters. Just for comparison, Figure 51 shows the original graph and the edge bundling layout produced by CUBu, the fastest technique in the state-of-the-art. Although CUBu generates the same graph in 1.437 seconds, the layout could not organize edges into meaningful bundles and the clutter reduction is not as effective as in the layout produced by our technique. Vertices in dense areas are equally hidden in CUBu layout as they are in the original one. Our technique needs around 100 seconds to create the backbone and 55 seconds to render the full graph, but created a less cluttered layout.

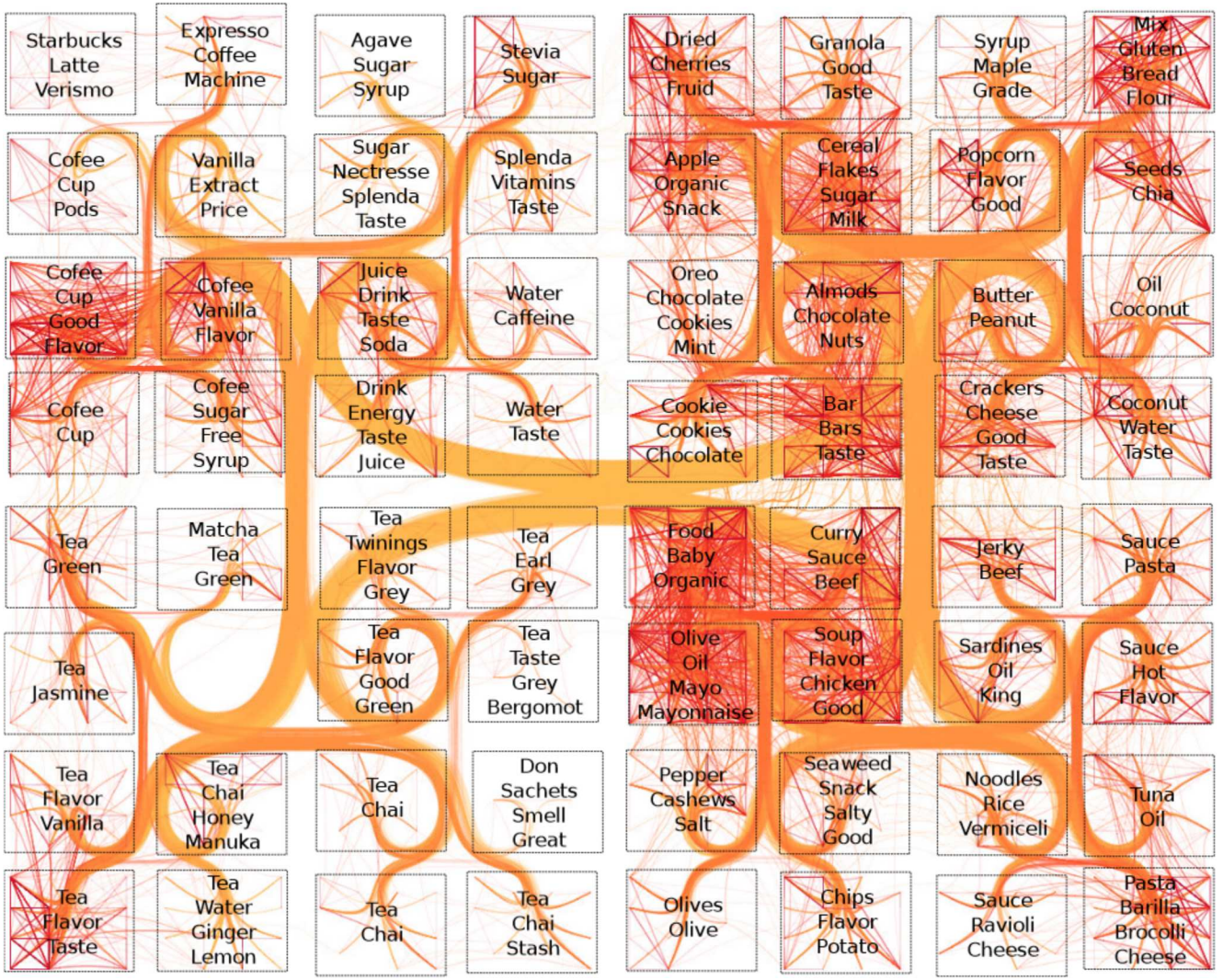
5.6 Final Remarks

In this chapter, we presented examples of technique developed during this master's research using graphs from different data sources. These applications, combined with the evaluation presented in the last chapter, provide enough evidence of the contribution of this research in the state-of-the-art.

Regarding large graphs, we believe that the fact that prior techniques perform parallelized tasks that can handle a large amount of data does not mean they can produce a good edge bundling layout. Instead, we take advantage of our multi-level similarity-based hierarchy to combine the edge bundling with a vertices filtering that reduces the amount of information, while grouping the most similar vertices into clusters.

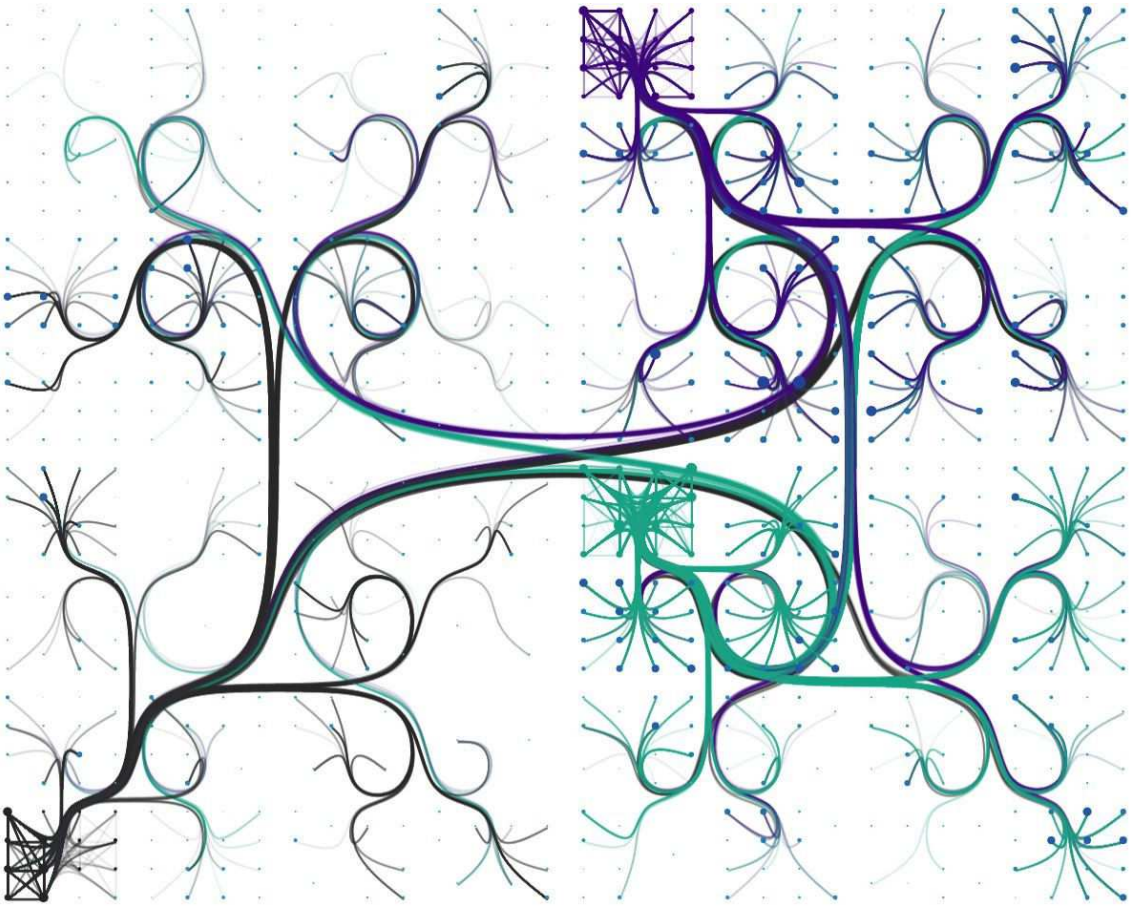
In the next chapter, we conclude this document with a critical review of our work, highlighting the contributions and discussing limitations and possible ideas for future work.

Figure 49 – SDEB layout of the dataset Amazon Groceries Reviews. The dataset has 8,700 vertices and 129,407 eges. However, this visualization is limited at the 9th level. The filtered graph has 843 vertices and 112,736 edges. Vertices were divided into 63 groups and each group is highlighted with its most common topics, extracted from the set of reviews written by customers.



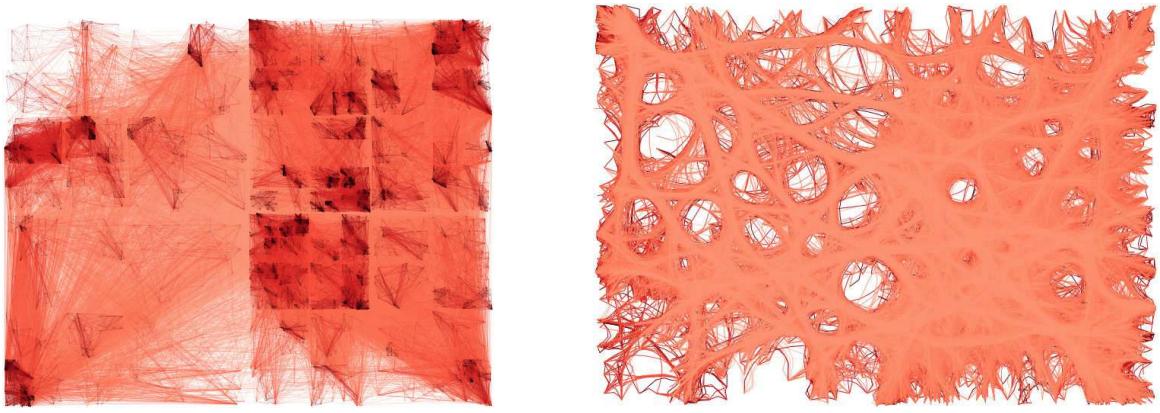
Source: Elaborated by the author.

Figure 50 – SDEB layout of the dataset Amazon Groceries Reviews with edges from 3 groups highlighted



Source: Elaborated by the author.

Figure 51 – Original and CUBu layouts of the dataset Amazon Groceries Reviews



(a) Original Graph (b) CUBu layout

Source: Elaborated by the author.

CHAPTER

6

CONCLUSION

In this chapter, we present the final remarks about this master's thesis, which consist of a review of the results, contributions, and suggestions of directions for future work. This research presented a novel edge bundling technique devised from two main principles: meaningful bundles can be created when similarity relationship are used to aggregate edges and a multiscale representation can improve the readability of bundling layouts for huge datasets.

Edge bundling techniques are a consolidated group of methods to construct the visualization of graphs. Several contributions have been made in this area in the last ten years. The state-of-the-art shows that grouping edges through curved bundles reduces the visual-clutter and improves the identification of edge patterns, while there is a trade-off between detailed and generic information. For example, edge bundling does not allow the identification of a single connection, but it may help global analytical studies in a graph. For such scenario, edge bundling techniques have achieved successful results.

This master's research added a new paradigm that highlights the relationship among vertices in order to create meaningful bundles. Our main contribution is a new technique that uses the data, instead of spatially information, to determine the bundles. We also validated this method using a broader evaluation, instead of a simple performance comparison and informal user studies, which have been done in most recent publications.

Different from contemporary edge bundling methods that rely on complex computational resources to achieve the shortest running time, like paralleled GPU algorithms, our technique uses straightforward algorithms from clustering and tree drawing. It makes our method easier to be replicated than the most recent ones. We are aware that our process can be further improved to reduce its running time, but this was not the goal of this study. In the next section, we detail a list of our contributions, followed by the discussions and suggestions for future work.

6.1 Contributions

The contribution of this research is a new edge bundling technique, called Similarity-driven Edge Bundling. This technique is the result of a bundling framework that consists of two main steps. In the first one, we create a structure called backbone from a series of studies on similarity based methods, which works as a hierarchy. Then, the backbone is used to bend the edges through similar routes, which creates edge bundles, thus reducing the visual-clutter in the graph layout.

Most of the previous techniques did not consider the similarities of bundled edges and, consequently, produced less meaningful representations. Although a few recent methods have embraced this discussion, they only adapted former techniques. For the best of our knowledge, this research presents the first technique that use exclusively a similarity-based approach. Our method may lead to new discussions and contributions for future research on edge bundling.

Furthermore, we performed a robust evaluation of our technique. The backbone evaluation validated the precision in which the backbone describes the similarity among elements. This evaluation also verified that our method produces more balanced branches, which creates less distorted bundles. Later on, we used artificial datasets to show that our technique reproduces known patterns contained on the data. Since the lack of evaluation is a problem in the field, our evaluation framework is also a contribution to this discussion.

Moreover, we presented a novel bundling model for large graphs, which consists of a multiscale bundling visualization. In former edge bundling techniques, the discussion of applications in large graphs only considered performance issues and not the final layout. Even though some techniques can generate edge bundling layouts of large graphs in a short time, we believe that aspects of their visualization, such as the limited visual space and vertices overlapping, make those useless without data filtering. Our backbone method produces a better visualization by grouping similar vertices and keeping the visual space clearer. We presented applications (see chapter 5) that confirmed the multiscale bundling ability of extracting information from large datasets.

Finally, we also presented a straightforward derivation of our technique to handle dynamic graphs with a fixed set of vertices. In this case, we built the backbone from vertices similarities and then processed a time series of edges. Most edge bundling techniques were initially developed for static graphs and then adapted for some dynamic scenarios. Our approach can be applied to dynamic graphs without any significant change. In addition, we guarantee the stability from the time-varying set of edges, i.e., our technique will render an edge in the same way in any time frame. This is different from the density-based methods, which may render the same edge in different ways depending on the other edges.

6.2 Limitations and Future Work

We outline the following points as limitations of our work and briefly discuss directions that can be taken to fill these gaps:

- Similarity based approaches for fixed vertices layouts: The main limitation of this research is the inability to bundle edges for graph layouts when the vertices have fixed positions. This limitation affects the comparison with most state-of-the-art techniques. Our backbone process can not handle properly this kind of graph because it might not place the intermediate vertices in a suitable form to create pleasant bundles. A possible solution could be to change the tree-like backbone for a similarity-based grid.

- Edge bundling evaluation: This research presented a more extensive evaluation process than many prior techniques, but we also relied on a subjective pattern analysis. Edge bundling evaluation methods lack richer designed user studies and quantitative measures. User-experience tests are an essential task in information visualization, but they require many resources to perform useful and reliable tests, while poor designed ones might present biased results. For quantitative measures, the recent stress-based metric proposed by Nguyen, Eades and Hong (2013b) represents a large step in this direction. However, we could not determine a reliable edges compatibility measure to use in our evaluation. A future study that formulates better edges compatibility and curve fitting measures can turn this metric suitable for bundling evaluations.

- Bundling layout presentation: We presented some visual enhancement options for our technique, such as the transformed tension (Adaptive-β) and the intermediate vertices filtering. However, there are other options that were not explored in this research and could be used for ours and other edge bundling methods. Specially, coloring edges has been used in many techniques to improve the edge bundling graph readability and could be used to separate better the bundles or clusters of vertices.

- Edge bundling for dynamic graphs: Our technique can handle one particular scenario of dynamic graphs, since we require a fixed set of vertices to achieve a stable backbone. Any change in this set would lead to a loss of context. Edge bundling for dynamic graphs is an open field in visualization, and there is still a need for future research addressing edge bundling methods that can show complex time-varying changes and keep the context for edges and vertices.

6.3 Publications

The contributions of this research are reported in the following research papers:

- Sikansi, F., Silva R. R. O., Paulovich, F. V. Similarity-driven Edge Bundling: Revisiting Hierarchical Edge Bundling for semantic meaningful clutter reduction in graphs layouts. Manuscript in preparation

- Sikansi, F., Paulovich, F. V. (2015). Using phylogenetic trees to generate semantic meaningful edge bundles. In Conference on Graphics, Patterns and Images, XXVIII; Workshop on Visual Analytics, Information Visualization and Scientific Visualization, VI. Sociedade Brasileira de Computação-SBC. Short-paper

In addition, the following publications were developed in collaboration with other researchers during the development of this research:

- Duarte, F. S. L. D., Sikansi, F., Fatore, F. M., Fadel, S. G., Paulovich, F. V. (2014). Nmap: A novel neighborhood preservation space-filling algorithm. IEEE transactions on visualization and computer graphics, 20(12), (pp. 2063-2071).

- Ono, J. H. P., Sikansi, F., Corrêa, D. C., Paulovich, F. V., Paiva, A., Nonato, L. G. (2015, August). Concentric RadViz: visual exploration of multi-task classification. In 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images (pp. 165-172). IEEE. Computer Graphics/Visualization Honorable Mention Award

- Neves, T. T. A. T., Coimbra, D., Sikansi, F., Paulovich, F. V. A Single-pass Model for Multidimensional Projection and its Applications to Data Streaming. Manuscript in preparation

BIBLIOGRAPHY

ALEXANDERSON, G. About the cover: Euler and königsberg's bridges: A historical view. Bulletin of the american mathematical society, v. 43, n. 4, p. 567–573, 2006. Cited on page 1.

ALSAKRAN, J.; CHEN, Y.; LUO, D.; ZHAO, Y.; YANG, J.; DOU, W.; LIU, S. et al. Real-time visualization of streaming text with a force-based dynamic system. IEEE Computer Graphics and Applications, v. 32, n. 1, p. 34–45, 2012. Cited on page 17.

ARCHAMBAULT, D.; PURCHASE, H. C.; PINAUD, B. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 17, n. 4, p. 539–552, 2011. Cited on page 75.

ASUNCION, A.; NEWMAN, D. UCI Machine Learning Repository. 2007. Available: <http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html>. Cited on page 48.

BACH, B.; RICHE, N. H.; HURTER, C.; MARRIOTT, K.; DWYER, T. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. IEEE Transactions on Visualization and Computer Graphics, IEEE, 2016. Cited on page 11.

BATTISTA, G. D.; EADES, P.; TAMASSIA, R.; TOLLIS, I. G. Graph Drawing: Algorithms for the Visualization of Graphs. [S.l.]: Prentice Hall PTR, 1998. Cited on page 2.

BECK, F.; BURCH, M.; DIEHL, S.; WEISKOPF, D. The state of the art in visualizing dynamic graphs. EuroVis STAR, 2014. Cited on page 2.

BEN-AKIVA, M.; PALMA, A. D.; ISAM, K. Dynamic network models and driver information systems. Transportation Research Part A: General, Elsevier, v. 25, n. 5, p. 251–266, 1991. Cited on page 1.

BENNETT, C.; RYALL, J.; SPALTEHOLZ, L.; GOOCH, A. The aesthetics of graph visualization. In: CITESEER. Computational Aesthetics. [S.l.], 2007. p. 57–64. Cited on page 9.

BOYANDIN, I.; BERTINI, E.; LALANNE, D. A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples. In: WILEY ONLINE LIBRARY. Computer Graphics Forum. [S.l.], 2012. v. 31, n. 3pt2, p. 1005–1014. Cited on page 75.

BRULS, M.; HUIZING, K.; WIJK, J. J. V. Squarified treemaps. [S.l.]: Springer, 2000. Cited on page 11.

BURCH, M.; KONEVTSOVA, N.; HEINRICH, J.; HOEFERLIN, M.; WEISKOPF, D. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 17, n. 12, p. 2440–2448, 2011. Cited on page 10.

CUADROS, A. M.; PAULOVICH, F. V.; MINGHIM, R.; TELLES, G. P. Point placement by phylogenetic trees and its application to visual analysis of document collections. In: IEEE VAST. [S.l.: s.n.], 2007. p. 99–106. Cited on page 30.

CUI, W.; QU, H. A survey on graph visualization. PhD Qualifying Exam (PQE) Report, Computer Science Department, Hong Kong University of Science and Technology, Kowloon, Hong Kong, Citeseer, 2007. Cited on page 2.

CUI, W.; ZHOU, H.; QU, H.; WONG, P. C.; LI, X. Geometry-based edge clustering for graph visualization. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 14, n. 6, p. 1277–1284, 2008. Cited 4 times on pages 3, 10, 14, and 15.

DOYE, J. P.; MASSEN, C. P. Characterizing the network topology of the energy landscapes of atomic clusters. The Journal of chemical physics, AIP Publishing, v. 122, n. 8, p. 084105, 2005. Cited on page 1.

DWYER, T.; RICHE, N. H.; MARRIOTT, K.; MEARS, C. Edge compression techniques for visualization of dense directed graphs. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 19, n. 12, p. 2596–2605, 2013. Cited on page 2.

EADES, P. Drawing free trees. [S.l.]: International Institute for Advanced Study of Social Information Science, Fujitsu Limited, 1991. Cited on page 9.

ELLIS, G.; DIX, A. A taxonomy of clutter reduction for information visualisation. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 13, n. 6, p. 1216–1223, 2007. Cited 2 times on pages 2 and 10.

ELZEN, S. van den; HOLTEN, D.; BLAAS, J.; WIJK, J. J. van. Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks. In: IEEE. Visualization Symposium (PacificVis), 2013 IEEE Pacific. [S.l.], 2013. p. 33–40. Cited on page 1.

EPPSTEIN, D.; GOODRICH, M. T.; MENG, J. Y. Confluent layered drawings. Algorithmica, Springer, v. 47, n. 4, p. 439–452, 2007. Cited on page 2.

ERSOY, O. Image-based graph visualization. Phd Thesis (PhD Thesis) — University of Groningen, 2013. Cited on page 11.

ERSOY, O.; HURTER, C.; PAULOVICH, F. V.; CANTAREIRO, G.; TELEA, A. Skeleton-based edge bundling for graph visualization. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 17, n. 12, p. 2364–2373, 2011. Cited 8 times on pages 3, 4, 10, 19, 20, 21, 47, and 70.

FADEL, S. G. Understanding interactive multidimensional projections. Master's Thesis (Master's Thesis) — University of São Paulo, 2016. Cited on page 11.

FADEL, S. G.; FATORE, F. M.; DUARTE, F. S.; PAULOVICH, F. V. Loch: A neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. Neurocomputing, Elsevier, v. 150, p. 546–556, 2015. Cited on page 48.

FEKETE, J.-D.; GRINSTEIN, G.; PLAISANT, C. IEEE InfoVis 2004 Contest, the history of InfoVis. 2004. Http://www.cs.umd.edu/hcil/iv04contest. Cited 2 times on pages 30 and 69.

FINKEL, R. A.; BENTLEY, J. L. Quad trees a data structure for retrieval on composite keys. Acta informatica, Springer, v. 4, n. 1, p. 1–9, 1974. Cited on page 15.

GANSNER, E. R.; HU, Y.; NORTH, S.; SCHEIDEGGER, C. Multilevel agglomerative edge bundling for visualizing large graphs. In: IEEE. Pacific Visualization Symposium (PacificVis), 2011 IEEE. [S.l.], 2011. p. 187–194. Cited 3 times on pages 14, 16, and 54.

GANSNER, E. R.; KOREN, Y. Improved circular layouts. In: ____. Graph Drawing: 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 386–398. ISBN 978-3-540-70904-6. Available: <http://dx.doi.org/10.1007/978-3-540-70904-6_37>. Cited on page 16.

GOU, L.; YOU, F.; GUO, J.; WU, L.; ZHANG, X. L. Sfviz: Interest-based friends exploration and recommendation in social networks. In: Proceedings of the 2011 Visual Information Communication - International Symposium. New York, NY, USA: ACM, 2011. (VINCI '11), p. 15:1–15:10. ISBN 978-1-4503-0786-4. Available: <http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/2016656.2016671>. Cited on page 13.

GRAHAM, R. L.; HELL, P. On the history of the minimum spanning tree problem. IEEE Annals of the History of Computing, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 7, n. 1, p. 43–57, Jan. 1985. ISSN 1058-6180. Cited on page 29.

GROSS, J.; YELLEN, J. Graph Theory and Its Applications, Second Edition. [S.l.]: Taylor & Francis, 2005. (Textbooks in Mathematics). ISBN 9781584885054. Cited on page 7.

GUO, L.; ZUO, W.; PENG, T.; ADHIKARI, B. K. Attribute-based edge bundling for visualizing social networks. Physica A: Statistical Mechanics and its Applications, Elsevier, v. 438, p. 48–55, 2015. Cited on page 3.

HEARN, D.; BAKER, M. P. Computer Graphics. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1986. ISBN 0-13-165382-2. Cited on page 12.

HENRY, N.; BEZERIANOS, A.; FEKETE, J.-D. Improving the readability of clustered social networks using node duplication. IEEE Transactions on Visualization and Computer Graphics, IEEE, v. 14, n. 6, p. 1317–1324, 2008. Cited on page 2.

HERMAN, I.; MELANÇON, G.; MARSHALL, M. S. Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 6, n. 1, p. 24–43, 2000. Cited 2 times on pages 9 and 10.

HOLTEN, D. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 12, n. 5, p. 741–748, 2006. Cited 9 times on pages 13, 2, 11, 12, 27, 42, 44, 45, and 47.

HOLTEN, D.; CORNELISSEN, B.; WIJK, J. J. V. Trace visualization using hierarchical edge bundles and massive sequence views. In: IEEE. Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on. [S.l.], 2007. p. 47–54. Cited 2 times on pages 3 and 13.

HOLTEN, D.; WIJK, J. J. V. Force-directed edge bundling for graph visualization. In: WILEY ONLINE LIBRARY. Computer Graphics Forum. [S.l.], 2009. v. 28, n. 3, p. 983–990. Cited 4 times on pages 3, 10, 17, and 18.

HURTER, C.; ERSOY, O.; FABRIKANT, S. I.; KLEIN, T. R.; TELEA, A. C. Bundled visualization of dynamic graph and trail data. IEEE transactions on visualization and computer graphics, v. 20, n. 8, p. 1141–1157, October 2013. ISSN 1941-0506. Cited 2 times on pages 3 and 4.

HURTER, C.; ERSOY, O.; TELEA, A. Graph bundling by kernel density estimation. In: WILEY ONLINE LIBRARY. Computer Graphics Forum. [S.l.], 2012. v. 31, n. 3pt1, p. 865–874. Cited 5 times on pages 3, 20, 21, 22, and 47.

ISENBERG, P.; HEIMERL, F.; KOCH, S.; ISENBERG, T.; XU, P.; STOLPER, C.; SEDLMAIR, M.; CHEN, J.; MöLLER, T.; STASKO, J. Visualization Publication Dataset. 2015. Dataset: http://vispubdata.org/. Published Jun. 2015. Available: <http://vispubdata.org/>. Cited 3 times on pages 32, 55, and 69.

JAIN, A. K.; DUBES, R. C. Algorithms for clustering data. [S.l.]: Prentice-Hall, Inc., 1988. Cited 3 times on pages 3, 31, and 34.

JIA, Y.; HOBEROCK, J.; GARLAND, M.; HART, J. On the visualization of social and other scale-free networks. IEEE transactions on visualization and computer graphics, IEEE, v. 14, n. 6, p. 1285–1292, 2008. Cited on page 2.

JOHNSON, B.; SHNEIDERMAN, B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In: IEEE. Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on. [S.l.], 1991. p. 284–291. Cited on page 11.

KAUFMANN, M.; WAGNER, D. Drawing graphs: methods and models. [S.l.]: Springer, 2003. Cited on page 9.

KEIM, D. A. Information visualization and visual data mining. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 8, n. 1, p. 1–8, 2002. Cited on page 1.

KIENREICH, W.; SEIFERT, C. An application of edge bundling techniques to the visualization of media analysis results. In: IEEE. 2010 14th International Conference Information Visualisation. [S.l.], 2010. p. 375–380. Cited 2 times on pages 11 and 23.

KIKUCHI, S.; TOMINAGA, D.; ARITA, M.; TAKAHASHI, K.; TOMITA, M. Dynamic modeling of genetic networks using genetic algorithm and s-system. Bioinformatics, Oxford Univ Press, v. 19, n. 5, p. 643–650, 2003. Cited on page 1.

KNUTH, D. E. The Stanford GraphBase: a platform for combinatorial computing. [S.l.]: Addison-Wesley Reading, 1993. Cited on page 8.

KOFFKA, K. Principles of Gestalt psychology. [S.l.]: Routledge, 2013. Cited on page 27.

KRUSKAL, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, Springer-Verlag, v. 29, n. 1, p. 1–27, 1964. ISSN 0033-3123. Available: <http://dx.doi.org/10.1007/BF02289565>. Cited on page 53.

LAMBERT, A.; BOURQUI, R.; AUBER, D. Winding roads: Routing edges into bundles. In: WILEY ONLINE LIBRARY. Computer Graphics Forum. [S.l.], 2010. v. 29, n. 3, p. 853–862. Cited 3 times on pages 3, 14, and 15.

LANDESBERGER, T. V.; KUIJPER, A.; SCHRECK, T.; KOHLHAMMER, J.; WIJK, J. J. van; FEKETE, J.-D.; FELLNER, D. W. Visual analysis of large graphs: State-of-the-art and future research challenges. In: WILEY ONLINE LIBRARY. Computer graphics forum. [S.l.], 2011. v. 30, n. 6, p. 1719–1749. Cited 3 times on pages 2, 9, and 10.

LEMEY, P.; SALEMI, M.; VANDAMME, A. The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing. Cambridge University Press, 2009. ISBN 9781139478618. Available: <https://books.google.com.br/books?id=C47QjT2XEY0C>. Cited on page 29.

LHUILLIER, A.; HURTER, C. Bundling, simplification de graphes par agrégation visuelle: Etat de l'art et défis. In: ACM. 27ème conférence francophone sur l'Interaction Homme-Machine. [S.l.], 2015. p. a10. Cited on page 24.

MANSMANN, F.; FISCHER, F.; KEIM, D. A.; NORTH, S. C. Visual support for analyzing network traffic and intrusion detection events using treemap and graph representations. In: Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology. New York, NY, USA: ACM, 2009. (CHiMiT '09), p. 3:19–3:28. ISBN 978-1-60558-572-7. Available: <http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/1641587.1641590>. Cited on page 13.

MARTINS, R. M.; ANDERY, G. F.; HEBERLE, H.; PAULOVICH, F. V.; LOPES, A. de A.; PEDRINI, H.; MINGHIM, R. Multidimensional projections for visual analysis of social networks. Journal of Computer Science and Technology, Springer, v. 27, n. 4, p. 791–810, 2012. Cited on page 10.

MCAULEY, J.; LESKOVEC, J. Hidden factors and hidden topics: understanding rating dimensions with review text. In: ACM. Proceedings of the 7th ACM conference on Recommender systems. [S.l.], 2013. p. 165–172. Cited 2 times on pages 70 and 77.

MELANÇON, G.; HERMAN, I. Circular drawings of rooted trees. In: IN REPORTS OF THE CENTRE FOR MATHEMATICS AND COMPUTER SCIENCES. [S.l.: s.n.], 1998. Cited 2 times on pages 9 and 10.

MERCKEN, L.; SNIJDERS, T. A.; STEGLICH, C.; VARTIAINEN, E.; VRIES, H. D. Dynamics of adolescent friendship networks and smoking behavior. Social Networks, Elsevier, v. 32, n. 1, p. 72–81, 2010. Cited on page 1.

MUNZNER, T. Visualization Analysis and Design. 1. ed. [S.l.]: A K Peters/CRC Press, 2014. (AK Peters Visualization Series). ISBN 1466508914,9781466508910. Cited on page 2.

NGUYEN, Q.; EADES, P.; HONG, S.-H. On the faithfulness of graph visualizations. In: IEEE. Visualization Symposium (PacificVis), 2013 IEEE Pacific. [S.l.], 2013. p. 209–216. Cited 3 times on pages 4, 24, and 47.

____. Streameb: Stream edge bundling. In: SPRINGER. Graph Drawing. [S.l.], 2013. p. 400–413. Cited 3 times on pages 3, 54, and 85.

NGUYEN, Q.; HONG, S.-H.; EADES, P. Tgi-eb: A new framework for edge bundling integrating topology, geometry and importance. In: SPRINGER. Graph Drawing. [S.l.], 2012. p. 123–135. Cited 3 times on pages 3, 19, and 23.

NOCAJ, A.; BRANDES, U. Stub bundling and confluent spirals for geographic networks. In: SPRINGER. International Symposium on Graph Drawing. [S.l.], 2013. p. 388–399. Cited on page 1.

PAIVA, J. G.; FLORIAN, L.; PEDRINI, H.; TELLES, G.; MINGHIM, R. Improved similarity trees and their application to visual data classification. IEEE Transactions on Visualization and Computer Graphics, v. 17, n. 12, p. 2459–2468, Dec 2011. Cited on page 30.

PAULOVICH, F. V.; NONATO, L. G.; MINGHIM, R.; LEVKOWITZ, H. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 14, n. 3, p. 564–575, 2008. Cited on page 70.

PEYSAKHOVICH, V.; HURTER, C.; TELEA, A. Attribute-driven edge bundling for general graphs with applications in trail analysis. In: IEEE. Visualization Symposium (PacificVis), 2015 IEEE Pacific. [S.l.], 2015. Cited 3 times on pages 3, 23, and 24.

QU, H.; ZHOU, H.; WU, Y. Controllable and progressive edge clustering for large networks. In: SPRINGER. Graph Drawing. [S.l.], 2007. p. 399–404. Cited on page 14.

REINGOLD, E. M.; TILFORD, J. S. Tidier drawings of trees. Software Engineering, IEEE Transactions on, IEEE, n. 2, p. 223–228, 1981. Cited on page 9.

SAGA, R.; YAMASHITA, T. Multi-type edge bundling in force-directed layout and evaluation. Procedia Computer Science, Elsevier, v. 60, p. 1763–1771, 2015. Cited on page 19.

SAITOU, N.; NEI, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular biology and evolution, SMBE, v. 4, n. 4, p. 406–425, 1987. Cited 4 times on pages 3, 29, 30, and 51.

SALTON, G. Developments in automatic text retrieval. Science, The American Association for the Advancement of Science, v. 253, n. 5023, p. 974, 1991. Cited on page 70.

SELASSIE, D.; HELLER, B.; HEER, J. Divided edge bundling for directional network data. Visualization and Computer Graphics, IEEE Transactions on, IEEE, v. 17, n. 12, p. 2354–2363, 2011. Cited 4 times on pages 17, 18, 19, and 44.

SERRANO, M. A.; BOGUÑÁ, M. Topology of the world trade web. Physical Review E, APS, v. 68, n. 1, p. 015101, 2003. Cited on page 1.

SHILOACH, Y. Arrangements of Planar Graphs on the Planar Lattices. Phd Thesis (PhD Thesis) — Weizmann Institute of Science, Rehovot, Israel, 1976. Cited 2 times on pages 34 and 36.

SIKANSI, F.; PAULOVICH, F. V. Using phylogenetic trees to generate semantic meaningful edge bundles. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO-SBC. Conference on Graphics, Patterns and Images, XXVIII; Workshop on Visual Analytics, Information Visualization and Scientific Visualization, VI. [S.l.], 2015. Cited 2 times on pages 41 and 55.

SILVA, R. R. O. da; PAIVA, J. G. de S.; TELLES, G. P.; ROLLI, F.; ZAMPIERI, C. E. A.; MINGHIM, R. The Visual Supertree: Similarity-based multiscale visualization. Submitted to Information Visualization (IVI). 2016. Cited on page 31.

SOKAL, R. R.; MICHENER, C. D. A statistical method for evaluating systematic relationships. University of Kansas Scientific Bulletin, v. 28, p. 1409–1438, 1958. Cited 3 times on pages 3, 29, and 51.

SPEARMAN, C. The proof and measurement of association between two things. American Journal of Psychology, v. 15, p. 88–103, 1904. Cited on page 53.

SPENCER, R. Information Visualization: Design for Interaction. [S.l.]: Person Education,, 2007. Cited on page 2.

STERNBERG, R. J. Cognitive Psychology. [S.l.]: Cengage Learning, 2008. ISBN 9780495506294. Cited on page 27.

STUDIER, J. A.; KEPPLER, K. J. et al. A note on the neighbor-joining algorithm of saitou and nei. Molecular biology and evolution, v. 5, n. 6, p. 729–731, 1988. Cited on page 30.

SUN, G.-D.; WU, Y.-C.; LIANG, R.-H.; LIU, S.-X. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. Journal of Computer Science and Technology, Springer, v. 28, n. 5, p. 852–867, 2013. Cited on page 11.

SUN, M.; MI, P.; NORTH, C.; RAMAKRISHNAN, N. Biset: Semantic edge bundling with biclusters for sensemaking. IEEE transactions on visualization and computer graphics, IEEE, v. 22, n. 1, p. 310–319, 2016. Cited on page 3.

TAMASSIA, R. Handbook of graph drawing and visualization. [S.l.]: CRC press, 2013. Cited on page 36.

TAYLOR, T.; PATERSON, D.; GLANFIELD, J.; GATES, C.; BROOKS, S.; MCHUGH, J. Flovis: Flow visualization system. In: IEEE. Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology. [S.l.], 2009. p. 186–198. Cited on page 13.

TEJADA, E.; MINGHIM, R.; NONATO, L. G. On improved projection techniques to support visual exploration of multi-dimensional data sets. Information Visualization, SAGE Publications, v. 2, n. 4, p. 218–231, 2003. Cited on page 17.

TELEA, A.; ERSOY, O. Image-based edge bundles: Simplified visualization of large graphs. In: WILEY ONLINE LIBRARY. Computer Graphics Forum. [S.l.], 2010. v. 29, n. 3, p. 843–852. Cited 4 times on pages 3, 19, 20, and 44.

TELEA, A. C. Data Visualization : Principles and Practice. [S.l.]: CRC Press, 2007. ISBN 978-1-4398-6593-4,1439865930. Cited on page 2.

THULASIRAMAN, K.; SWAMY, M. N. Graphs: theory and algorithms. [S.l.]: John Wiley & Sons, 2011. Cited on page 8.

VORONOÏ, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. Journal für die reine und angewandte Mathematik, v. 134, p. 198–287, 1908. Cited on page 15.

WARD, M.; GRINSTEIN, G.; KEIM, D. Interactive data visualization: foundations, techniques, and applications. [S.l.]: AK Peters, Ltd., 2010. Cited on page 9.

WEISS, S. M. On the performance of bisecting k-means and pddp. SIAM, 2001. Cited on page 33.

YAMASHITA, T.; SAGA, R. Edge bundling in multi-attributed graphs. In: YAMAMOTO, S. (Ed.). Human Interface and the Management of Information. Information and Knowledge Design. Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9172). p. 138–147. ISBN 978-3-319-20611-0. Available: <http://dx.doi.org/10.1007/978-3-319-20612-7_14>. Cited 3 times on pages 3, 19, and 23.

ZWAN, M. van der; CODREANU, V.; TELEA, A. Cubu: Universal real-time bundling for large graphs. IEEE Transactions on Visualization and Computer Graphics, PP, n. 99, p. 1–1, 2016. ISSN 1077-2626. Cited 4 times on pages 3, 20, 22, and 23.