Applications of Multidimensional Scaling to Graph Drawing

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
an der Universität Konstanz
Fachbereich Informatik und Informationswissenschaft

vorgelegt von

Christian Pich

Tag der mündlichen Prüfung: 7. Juli 2009
Referent: Prof. Dr. Ulrik Brandes
Referent: Prof. Dr. Michael Berthold

Teile dieser Arbeit sind bereits veröffentlicht worden:

- Kapitel 3 (Brandes and Pich, 2007)
- Kapitel 5 (Brandes and Pich, 2009a)
- Kapitel 6 (Pich, 2009)
- Kapitel 7 (Pich et al., 2008)
- Kapitel 8 (Brandes and Pich, 2009b)

Zusammenfassung

Netzwerke sind in vielen Bereichen ein grundlegendes Modell, mit dem man Beziehungen zwischen bestimmten Objekten untersucht, etwa Personen und deren Bekanntschaften, Rechner im Internet, Interaktionen zwischen Proteinen oder Verkehrsnetze. Wegen der stets zunehmenden Bedeutung dieser Untersuchungen und der wachsenden Komplexität und Größe dieser Netzwerke wird auch die Visualisierung solcher Daten immer bedeutender.

Eine geeignete grafische Darstellung kann, neben Zwecken der Präsentation, auch zur Interpretation und einem besseren Verständnis der Eigenschaften eines betrachteten Netzwerks beitragen. Dies ist aber abhängig von bestimmten Anforderungen: Einerseits gibt es oft anwendungsspezifische ästhetische Vorgaben und Konventionen; andererseits sollte eine Darstellung die Struktur des Netzwerks, also den darunterliegenden Graphen, getreu wiedergeben und nicht unnötig verfälschen.

Für die Erstellung von Netzwerkvisualisierungen ist aus Sicht der Informatik vor allem die Wiedergabe der Struktur bedeutend. Im Forschungsgebiet des Graphenzeichnens werden Verfahren entwickelt, um automatisch Positionen für Objekte in einem Netzwerk zu bestimmen; es gibt eine Vielzahl verschiedener Modelle und Algorithmen, deren Auswahl stark vom Einsatzgebiet abhängt.

Ein Ansatz, der wegen seiner Einfachheit und Effektivität und seiner guten Ergebnisse häufig verfolgt wird, ist das kräfte- oder energiebasierte Zeichnen. Die geometrische Anordnung der Objekte in einem Graphen wird mit einem Gütekriterium bestimmt, das bewertet, wie gut Ähnlichkeit oder Nähe im Graphen in der Visualisierung wiedergegeben werden.

Einige in diesem Teilbereich erarbeitete Verfahren treten auch in anderen Forschungsgebieten auf, etwa der Datenanalyse, der Psychometrie, der Soziologie oder der Statistik. Obwohl die dort untersuchten und damit visualisierten Gegenstände nur vereinzelt Graphen sind, können viele der Verfahren mit Anpassungen auf Graphen angewandt werden und führen erfahrungsgemäß zu aussagekräftigen Visualisierungen.

Diese Arbeit befasst sich mit der Anwendung der Mehrdimensionalen Skalierung (MDS) auf das Zeichnen von Graphen. MDS ist eine Familie von Methoden

zur Analyse und Visualisierung von Ähnlichkeitsdaten. Wenn diese Methoden auf Graphen angewendet werden, ist ihr gemeinsames Hauptziel, dass Knoten, die im Graphen nahe beieinanderliegen, auch in der Zeichnung benachbart sind.

Einerseits verwenden einige der bekannten Zeichenverfahren MDS bereits implizit oder explizit. Andererseits werden manchmal wichtige Eigenschaften und Vorteile von MDS nicht beachtet. Teil I dieser Arbeit behandelt die methodischen Grundlagen und soll diese Lücke schließen.

Die Klassische Skalierung ist die älteste bekannte Variante von MDS und wird in Kapitel 3 auf die Abstände in Graphen angewendet. Unter der Annahme, dass diese ursprünglich von Positionen aus einem Euklidischen Raum stammen, werden Techniken aus der Linearen Algebra verwendet, um diese Positionen zu rekonstruieren und eindeutige, in einem bestimmten Sinne optimale Zeichnungen zu finden. Da die verwendeten Matrixoperationen nur bis zu bestimmten Eingabegrößen skalierbar sind, wird ein Verfahren vorgestellt, mit dem die Berechnung wesentlich beschleunigt und für große Graphen überhaupt erst möglich wird.

Der meistverwendete MDS-Ansatz ist Distanz-Skalierung. Dabei wird die Anordnung mit einer Zielfunktion bewertet, in welche die Graph- und Layoutdistanzen direkt eingehen; mit numerischen Optimierungsalgorithmen wird diese Anordnung iterativ verbessert. Kapitel 4 stellt die Beziehungen zu bekannten Zeichenalgorithmen her. Außerdem wird eine Erweiterung beschrieben, mit der Nebenbedingungen als zusätzliche Zielfunktion definiert werden.

In einer experimentellen Studie in Kapitel 5 werden aus Erfahrungen und allgemeinen Annahmen Hypothesen über verschiedene Varianten von MDS und andere Zeichenverfahren hergeleitet, zu denen Experimente z.B. mit Initialisierungen und Beschleunigungen ausgeführt werden. Es wird eine Kombination bestimmter Algorithmen vorgeschlagen, die, gestützt auf Ergebnisse der Experimente, die besten Lösungen verspricht.

In Kapitel 6 werden eine Methode und ein Linearzeitalgorithmus zum Zeichnen gerichteter Graphen vorgestellt, die eine Zerlegung schiefsymmetrischer Matrizen verwenden. Dabei sollen Kanten nach Möglichkeit als Kurve im Uhrzeigersinn um den Ursprung gezeichnet werden statt von oben nach unten.

Drei Anwendungen zeigen in Teil II der Arbeit, wie die vorgestellten Methoden an verschiedene Anforderungen angepasst und erweitert werden und illustrieren ihre Skalierbarkeit: Bei Abhängigkeiten zwischen Komponenten großer Softwaresysteme (Kapitel 7) wird MDS mit anderen Visualisierungsmethoden kombiniert. Besondere ästhetische Vorgaben werden beim radialen Layout sozialer Netzwerke als Nebenbedingungen ausgedrückt (Kapitel 8). Eine Methode zur Darstellung und Analyse großer bibliographischer Netzwerke (Kapitel 9) verwendet schließlich eine schnelle Variante von MDS für erweiterte Graphenmodelle.

Acknowledgements

There are so many people who have contributed to this thesis directly or indirectly and to which I am indebted. I have to apologize for not being able to name all of them.

First and foremost, I want to express my deep gratitude to my advisor Ulrik Brandes, for always being supportive, challenging, patient, encouraging, curious, and reliable. Working with him and having him as a teacher and mentor was a great pleasure to me and is an absolute privilege I cannot appreciate enough!

Many visits to conferences and workshops would not have been possible without the financial support by the University of Konstanz. Particularly, I acknowledge support by the graduate program *Explorative Analysis and Visualization of Large Information Spaces* funded by the Deutsche Forschungsgemeinschaft.

Work and research is no fun without great colleagues – I was happy to be part of a very stimulating environment of creative and bright people. Standing for all of those I had the pleasure to work with, I thank Christian Bachmaier, Melanie Badent, Sabine Cornelsen, Gabi Dorfmüller, Martin Hoefer, Natalie Indlekofer, and Barbara Pampel, for sharing the office with me; particularly, I would like to thank Thomas Schank for being my long-term office mate and for providing nibbles, oatflakes, and coffee beans.

Special credits go to Christine Agorastos, Sven Kosub, Martin Mader, Uwe Nagel, and Bobo Nick, for their valuable and helpful comments on earlier draft versions of some chapters of this thesis.

I thank Michael Berthold for being the second referee and examiner, and Daniel Keim for joining my examination committee.

I am much obliged to Lev Nachmanson and George Robertson for inviting me to Microsoft Research. The three months in Redmond were among the most productive and stimulating times of my research life, and when I came back home I had learned that graph drawing has a real practical impact!

Finally, I am very grateful to my family, especially to my parents, Helga and Ronald Pich, for their never-ending support in every possible way and for allowing me to become what I wanted. I want to thank my wife Helga for all her love, care, and patience. Without her, everything would be nothing.

Contents

1 Introduction **1**

2 Preliminaries **5**

I Methods **11**

3 Classical Scaling **13**
3.1 Basic Method 14
3.1.1 Euclidean Distances 14
3.1.2 Dissimilarities 16
3.1.3 Strain 17
3.1.4 Computation 18
3.2 Classical Scaling for Graphs 21
3.2.1 Graph Distances 21
3.2.2 Eigenvalues 23
3.3 Speedup 25
3.3.1 Pivoting 27
3.3.2 Pivot MDS 28
3.3.3 Landmark MDS 33
3.4 Related Methods 34
3.4.1 Principal Component Analysis 36
3.4.2 Unfolding 37
3.4.3 High Dimensional Embedding 38

4 Distance Scaling **41**
4.1 Stress Minimization 42
4.1.1 Stress 42
4.1.2 Weights 43
4.1.3 Stress Majorization 45
4.2 Related Methods 48

4.2.1 Barycenter Layout 49
4.2.2 Spectral Layout 49
4.2.3 Force-directed Layout 50
4.2.4 Energy-based Layout 51
4.3 Weak Constraints 51
4.3.1 A Modified Stress Model 53
4.3.2 Computation 53
4.3.3 Applications 54

5 Experimental Study 57
5.1 Hypotheses 58
5.2 Experimental Design 59
5.3 Experiments 61
5.4 Results 66
5.5 Conclusion 67

6 Skew-symmetric Scaling 73
6.1 Skew-Symmetry 74
6.1.1 Skew-Symmetric Matrices 74
6.1.2 Decomposition 75
6.1.3 Eigenvalues 76
6.1.4 Interpretation 77
6.1.5 Computation 79
6.2 Clockwise Drawing of Directed Graphs 80
6.2.1 Skew-Symmetric Adjacency Matrices 81
6.2.2 A Two-Dimensional Drawing Method 82
6.2.3 Computation 85
6.3 Application to Tournaments 86
6.4 Conclusion 89

II Applications 91

7 Visual Analysis of Software Dependencies 93
7.1 Dependency Graphs 94
7.2 Layout Algorithm 95
7.2.1 Horizontal Coordinates 95
7.2.2 Vertical Coordinates 96
7.2.3 Adding Group Information 99
7.3 Applications 100
7.3.1 Java Standard Package 100

7.3.2 A Microsoft Software Project . . . 101
7.4 Conclusion . . . 104

8 Radial Layout **109**
8.1 Stress, Weights, and Constraints . . . 110
8.1.1 Stress . . . 110
8.1.2 Weights for Constraints . . . 111
8.1.3 Interpolated Weights . . . 112
8.2 Applications . . . 113
8.2.1 Target Diagrams . . . 113
8.2.2 Centrality Drawings . . . 113
8.2.3 Travel Time Maps . . . 117
8.3 Conclusion . . . 118

9 Layout of Bibliographic Networks **121**
9.1 Bibliographic Data Model . . . 122
9.1.1 Objects . . . 122
9.1.2 Basic Relationships . . . 122
9.1.3 Bibliographic Operators . . . 123
9.1.4 Bibliographic Proximity . . . 124
9.2 A Bibliography of Social Network Analysis . . . 126
9.2.1 Web Of Science Queries . . . 126
9.2.2 Bibliographic Coupling Networks . . . 128
9.2.3 Visualization . . . 128
9.3 Conclusion . . . 129

10 Conclusion **135**

List of Figures **137**

List of Algorithms **139**

Bibliography **141**

Index **159**

Chapter 1

Introduction

Networks are fundamental in many areas of research as a model for studying relations between objects, such as persons and their social ties, computers in the Internet, interactions between proteins, or traffic systems. Due to the increasing importance of these studies and the steadily growing complexity of these networks, their visualization is increasingly relevant, as well.

Aside from mere presentation purposes, an appropriate visual representation is able to significantly contribute to insight into the structural properties of a network under study. This is subject to certain requirements: First, there are frequently context-specific aesthetic constraints and conventions. Second, the visualization is required to represent the network structure, i.e., the underlying graph, appropriately, and should not be misleading.

From a computer science point of view, it is the representation of the structure that is crucial for the construction of network visualizations. The field of graph drawing is concerned with methods for the automatic computation of geometric positions for objects in networks. There are countless models and algorithms, and the concrete choice is specific to the particular application.

An approach frequently used due to its simplicity, its effectivity, and its appealing results, is force- or energy-based drawing. The geometric arrangement of graph elements is determined using a quality measure, which assesses how well similarity of proximity in the graph is reflected by the visualization.

Some of the graph drawing methodologies have counterparts in other scientific disciplines, such as data analysis, psychometrics, sociology, or statistics. Although designed for more general data analysis tasks, many of these methods can be adapted to graphs and empirically lead to appealing and meaningful visualizations.

This thesis is concerned with the application of Multidimensional Scaling (MDS) to graph drawing. MDS is an entire family of methods for analyzing data about similarity or proximity. In the context of graphs, the principal objective

is that if nodes are proximate in the graph, they should also be proximate in the visual representation.

In some graph drawing applications this objective is not formulated explicitiy; and even if so, important advantageous features of MDS are sometimes not considered. It is the objective of this thesis to fill this conceptual gap.

The challenges of proximity approaches are three-fold: Conceptually, it has to be determined what proximity actually means; mathematically, a model has to be formulated that captures this intuition; algorithmically, methods have to be devised that are able to turn the one proximity into the other. This thesis is devoted to all of these aspects.

The mentioned general objective is explicitly formulated in terms of mathematical objective functions, which are sought to be minimized by MDS. This gives rise to some algorithmic methods that are elegant, robust, flexible, scalable, and easy to implement. This thesis is intended to present a unified view on these methods and their application to graph drawing and to illustrate their practical usefulness. It is organized in two main parts and structured as follows:

Chapter 2: Preliminaries. The fundamental notation and some essential definitions required throughout the entire thesis are given. Further concepts are introduced only where necessary.

Part I: Methods

Chapter 3: Classical Scaling. The earliest variant of multidimensional scaling uses elementary linear algebra and Euclidean geometry. It is based on the spectral decompositions and produces unique optimal results. We apply classical scaling to the distance matrices of graphs and present a novel method to significantly reduce the time and space complexity. Classical scaling is thus made applicable even to very large graphs, for which the original classical scaling and almost all other graph drawing methods are prohibitive.

Chapter 4: Distance scaling. A very popular and widely used graph drawing approach is based on a goal function, termed *energy* or *stress*, which measures how well the current layout corresponds to the desired distances. Graph layouts are computed by iteratively minimizing this goal function. This methodology has been known long before its application to graph drawing in more general contexts. The chapter discusses these connections and presents an extension in which additional constraints are integrated into the layout process.

Chapter 5: Experimental Study. An extensive experimental study about the quality of MDS and other graph layout methods based on graph-theoretical distances is presented. The principal result is a recommendation to use a specific combination of MDS methods and algorithms to efficiently obtain the visually most pleasing results. The experimental methodology is based on a set of hypotheses; the experiments are designed so as to support or reject these hypotheses.

Chapter 6: Skew-symmetric Scaling. Directed graphs are frequently drawn in a hierarchical downwards style. While this method is appropriate in most applications, cycles tend to become difficult to detect. In this chapter a little known variant of MDS is adapted to directed graphs and their clockwise layout. This method analyzes the purely asymmetric component of the information expressed by the asymmetric adjacency matrices of directed graphs. It is particularly useful for the drawing, in which the objective of drawing edges downwards is replaced by the objective of drawing it clockwise around the origin.

Part II: Applications

Chapter 7: Visual Analysis of Software Dependencies. A method for the visual display and analysis of large software systems is presented. The dependencies and hierarchies in components of software systems are modeled by dependency graphs. The visualization of these dependency graphs uses a combination of classical scaling and other network analysis methods based on linear algebra. The method integrates the joint analysis of importance and hierarchy information in the visualization.

Chapter 8: Radial Layout. Nodes of a graph are constrained to lie on the circumferences of a set of concentric circles around the origin. Such constraints frequently occur in the layouts of social or policy networks, or when the primary intention of the layout is the display of the distances from a distinguished node which is put in the focus. The radial constraints are imposed by dynamically modifying the weights in the objective function to be minimized.

Chapter 9: Layout of Bibliographic Networks. The fast pivot-based version of classical scaling is applied to the visualization of large bibliographic data sets and illustrates that MDS is applicable to large-scale problems. The visual analysis concentrates on similarity and proximity in multimode networks, which

are a model for the relationships in the bibliographic data, and which encompass authors, works, journals, and scientific disciplines, and the corresponding connections, such as citations, affiliations, and authorship.

Chapter 10: Conclusion. Finally, the main results and contributions of this thesis are summarized.

Some parts of this thesis have been previously published:

- Chapter 3 (Brandes and Pich, 2007)
- Chapter 5 (Brandes and Pich, 2009a)
- Chapter 6 (Pich, 2009)
- Chapter 7 (Pich et al., 2008)
- Chapter 8 (Brandes and Pich, 2009b)

Chapter 2

Preliminaries

This chapter contains notation and definitions of formal concepts occurring frequently in this thesis. Further definitions shall be given only where necessary. For more details, see, for example, Diestel (2006) about graph theory, Meyer (2001) and Golub and van Loan (1996) about linear algebra and matrix computations, and Cormen et al. (2001) about graph algorithms.

Graphs The fundamental object of interest in this work is a *graph* $G = (V, E)$, encompassing a set $V = V(G)$ of n *nodes* or *vertices* and a set $E = E(G)$ of m *edges* or *links*; $n, m \in \mathbb{N}$. For convenience, nodes are frequently indexed with numbers, i.e., $V = \{1, \dots, n\}$ or $V = \{v_1, \dots, v_n\}$.

A graph is *undirected* if all its edges are two-element subsets $\{u, v\} \subseteq V$, thus $E \subseteq \binom{V}{2}$; a graph is *directed* if edges are ordered node pairs (v, w) from the *source* v to the *target* w, thus $E \subseteq V \times V$. If $\{v, w\} \in E$, v and w are *adjacent* to each other, and *incident* to the edge $\{v, w\}$. Two edges $\{u, v\}, \{v, w\} \in E$ are called *incident in* v. These definitions for directed graphs are analogous. The *underlying undirected graph* of a directed graph is obtained by replacing each directed edge (v, w) with its undirected version $\{v, w\}$.

In a graph $G = (V, E)$ the set of nodes adjacent to v is called the set of *neighbors* or *neighborhood* of v and denoted $N(v)$. If G is directed, $N^-(v) = \{u \in V : (u, v) \in E\}$ is called *set of predecessors* or *incoming neighborhood* and $N^+(v) = \{w \in V : (v, w) \in E\}$ the *set of successors* or *outgoing neighborhood*. The undirected neighborhood is defined as $N(v) = N^-(v) \cup N^+(v)$. The cardinalities $\deg(v) = |N(v)|$, $\deg^-(v) = |N^-(v)|$, and $\deg^+(v) = |N^+(v)|$ are called *degree*, *indegree*, and *outdegree* of v.

The *edge-induced subgraph* $G' = (V, E')$ is obtained by retaining only the edges in a subset $E' \subseteq E$ and their incident nodes, and deleting all others, denoted by $G[E']$. The *(node-)induced subgraph* $G[V']$ is obtained by retaining a node subset $V' \subseteq V$ and only edges whose both incident nodes are in V'.

Distance A *path* is a sequence of incident edges $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ which *connects* v_0 and v_k, and is called v_0v_k-path, with $v_0, \ldots, v_k \in V$. The number $k \in \mathbb{N}$ of its edges is called *length* of the path. Directed paths are defined analogously.

An st-path of length k from source node $s \in V$ to target node $t \in V$ is called a *shortest (st-)path* if there is no st-path of length less than k. In this case, k is called *graph-theoretical distance* or *shortest-path distance* from s to t, written as $d(s,t) = d_{st} = k$; clearly, $d_{vv} = 0$ for all $v \in V$. The largest distance between any two nodes in G is called *diameter* $\operatorname{diam}(G)$ of G.

If no path connects nodes $v, w \in V$, v and w are *disconnected*, and $d_{vw} = \infty$. For a given node $v \in V$, the graph $G[\{w \in V \colon d(v,w) < \infty\}]$ is called *connected component* of v. G is called *connected* if it has only one connected component, and *disconnected* otherwise. In this thesis it is generally assumed that graphs are connected; otherwise the methods are applied to the connected components individually.

Weights A graph is called *weighted*, written $G = (V, E, w)$, if a function $w \colon E \to \mathbb{R}_{>0}$ assigns positive real values $w(e)$ to each edge $e \in E$. A graph without an explicit weight function can be interpreted as a weighted graph with uniform edge weights, i.e. $w(e) = 1$ for all $e \in E$. The concrete interpretation of weights is specific to the application at hand, e.g. length, reliability, strength, or similarity.

In weighted graphs the length of a path is defined as the sum of edge weights along this path; the definitions of degree, distance, and diameter are carried over straightforwardly.

Vectors A real (row) *vector* $x \in \mathbb{R}^n$ of length $n \in \mathbb{N}$ is an n-tuple of entries $[x_1, \ldots, x_n]$. Usually, vectors will be column vectors, and writing them in a line as a row vector requires *transposition* $x = [x_1 \ldots x_n]^T$. The vectors $[0, \ldots, 0]^T, [1, \ldots, 1]^T \in \mathbb{R}^n$ are denoted $0_n, 1_n$. An important operation defined on two vectors $x = [x_1 \ldots x_n]^T, y = [y_1 \ldots y_n]^T \in \mathbb{R}^n$ of equal length is the *inner product* $\langle x, y \rangle = x^T \cdot y = x_1y_1 + \cdots + x_ny_n$. If $\langle x, y \rangle = 0$, x and y are *orthogonal*. A vector $x \neq 0_n$ is *scaled* with a constant $c \in \mathbb{R}$ by the multiplication $cx = [cx_1, \ldots, cx_n]^T$; the division x/c is shorthand for $\frac{1}{c}x$.

The *norm* or *length* $\|x\|$ of a vector $x \in \mathbb{R}^n$ is defined as $\|x\| = \sqrt{\langle x, x \rangle} = \sqrt{x_1^2 + \cdots + x_n^2}$. If $\|x\| = 1$, x is a *unit (length)* or *normalized* vector. A vector is normalized by $x/\|x\|$.

Matrices A *matrix* $A = (a_{ij}) \in \mathbb{R}^{n \times m}$ has n rows (*height* n) and m columns (*width* m); A is called *square* matrix if $n = m$. Individual entries are denoted

a_{ij} using the corresponding lowercase character. The *transpose* $A^T \in \mathbb{R}^{m\times n}$ of a matrix $A \in \mathbb{R}^{n\times m}$ is obtained by flipping row and column indices. The vector $A_{i\bullet} = [a_{i1}, \ldots, a_{im}]$ is called ith *row (vector)* and the vector $A_{\bullet j} = [a_{1j}, \ldots, a_{nj}]^T$ is called jth *column (vector).*

A square matrix $A = \{a_{ij}\} \in \mathbb{R}^{n\times n}$ is called

- *symmetric* if $A = A^T$;
- *skew-symmetric* if $A^T = -A$;
- *diagonal* if only its main diagonal contains non-zero entries, i.e., if $a_{ij} = 0$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$;
- *Euclidean* if there are coordinates $x_1, \ldots, x_n \in \mathbb{R}^d$ in some d-dimensional Euclidean space such that $a_{ij} = \|x_i - x_j\|$ for all $i, j \in \{1, \ldots, n\}$;
- *orthogonal* if all columns are pairwise orthogonal.

The *(Euclidean) norm* $\|A\|$ of a matrix $A = (a_{ij}) = \mathbb{R}^{n\times m}$ is the square root of the sum of squares of its entries, $\|A\| = \sqrt{\sum_{i=1}^n \sum_{j=1}^m a_{ij}^2}$. When A is a square matrix, its *trace* $\mathrm{tr}A$ is the sum of diagonal entries, $\mathrm{tr}A = \sum_{i=1}^n a_{ii}$. The diagonal matrix corresponding to a vector $x \in \mathbb{R}^n$ is denoted $\mathrm{diag}(x) \in \mathbb{R}^{n\times n}$. A diagonal matrix with ones is called *identity matrix* $I_n = \mathrm{diag}(1_n)$.

Eigendecompositions Let $A \in \mathbb{R}^{n\times n}$ be symmetric. A vector $u \in \mathbb{R}^n$ with $u \neq [0, \ldots, 0]^T$ is called *eigenvector* associated with *eigenvalue* $\lambda \in \mathbb{R}$ if $Ax = \lambda x$. The pair (λ, u) is also called *eigenpair* of A. The set of (not necessarily distinct) eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R}$ is called the *spectrum* of A.

The *spectral theorem*, a basic result from linear algebra, states that the corresponding eigenvectors $u_1, \ldots, u_n \in \mathbb{R}^n$ can be chosen to be mutually orthogonal, $\langle u_i, u_j \rangle = 0$ for $i \neq j$, and form a basis of $\mathbb{R}^n$.

The *rank* of a symmetric matrix is defined as the number of non-zero eigenvalues. If A has only positive (negative) eigenvalues, it is *positive (negative) definite*, and *positive (negative) semi-definite* if it has only nonnegative (nonpositive) eigenvalues. A useful equality for a symmetric matrix $A = (a_{ij})$ is that of its trace and the sum of its eigenvalues, $\sum_{i=1}^n a_{ij} = \sum_{i=1}^n \lambda_i$.

The *spectral decomposition* allows A to be written as

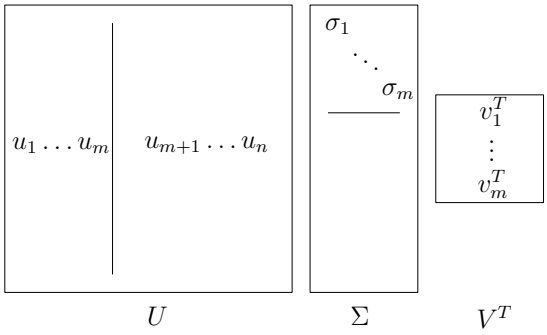
$$A = U\Lambda U^T = \underbrace{\begin{bmatrix} u_1 \ldots u_n \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} u_1^T \\ \vdots \\ u_n^T \end{bmatrix}}_{U^T}$$

such that $U = [u_1, \ldots, u_n]^T \in \mathbb{R}^{n \times n}$ contains the unit length eigenvectors of A and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. A may also be written as a sum of elementary rank-1 matrices

$$A = \sum_{i=1}^{n} \lambda_i \cdot u_i u_i^T.$$

A more general related decomposition exists for non-square matrices $B \in \mathbb{R}^{n \times m}$ (without loss of generality, $n \geq m$) into

$B = U\Sigma V^T =$



where, without loss of generality, $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n \times n}$ contains the unit-length *left singular vectors*, $V = [v_1, \ldots, v_m] \in \mathbb{R}^{m \times m}$ contains the unit-length *right singular vectors*, and $\Sigma \in \mathbb{R}^{n \times m}$ is a diagonal matrix of *singular values* $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$, with an additional $(n - m) \times m$ block of zero entries. This gives a sum of matrices

$$B = \sum_{i=1}^{m} \sigma_i \cdot u_i v_i^T$$

similar to the one for square matrices. The left singular vectors u_i are the eigenvectors of BB^T and the right singular vectors are the eigenvectors of $B^T B$, both corresponding to the eigenvalue σ_i^2.

Graphs and Matrices Instead of numbers $i, j \in \{1, \ldots, n\}$, the entries in graph-related vectors and matrices are frequently indexed by nodes as x_v for $v \in V$, and with pairs of nodes as a_{vw} for $v, w \in V$.

Several matrices can be associated with a graph. One of the basic ways of representing a graph is its *adjacency matrix* $A = (a_{uv})$, with $a_{uv} = 1$ if $\{u, v\} \in E$ ($(u, v) \in E$ for directed graphs), and $a_{uv} = 0$ otherwise; the adjacency matrices of weighted graphs contain the edge weights as entries instead of 1's. A basic ingredient for several methods in this thesis is the *distance matrix* $D = (d_{uv})$ of the graph-theoretical distances between nodes u and v.

Layout A *layout* of a graph $G = (V, E)$ is a *configuration* of d-dimensional positions $p(v) \in \mathbb{R}^d$ for all $v \in V$. In most applications $d = 2$, and positions will then be denoted as *coordinates* $p(v) = (x_v, y_v) \in \mathbb{R}^2$. Depending on the particular application and the analytic perspective, a *configuration matrix* $P \in \mathbb{R}^{n \times d}$ is written either in terms of the n position vectors as $P = [p(v_1), \ldots, p(v_n)]^T$, or in terms of n-dimensional vectors as $P = [P^{(1)}, \ldots, P^{(d)}]$.

Part I

Methods

Chapter 3

Classical Scaling

This chapter is devoted to the earliest practical approach of multidimensional scaling and its application to graph distances. The primary assumption behind classical scaling is that the input dissimilarities are derived from points in a *Euclidean space* whose coordinates are unknown, but can be recovered exactly or at least as well as possible, using elementary tools from linear algebra. The standard example is a road travel time table, which is a matrix of pairwise distances between some larger cities, whose original geographical positions can be recovered by classical scaling, at least up to reflection, rotation, and translation.

While these assumptions are reasonable in geographic applications, they often do not apply to other settings. This and the rise of computers for numerical optimization has led to the development of *nonmetric scaling* (see Chapter 4) in the 1960s, initiated by Shepard (1962) and Kruskal (1964), hence occasionally called *Kruskal-Shepard scaling*, and has since become the widely-used default MDS approach, while classical scaling seems to have fallen off favor.

In the field of graph drawing, the classical approach has long gone little noticed, as well. Only recently, it came back to attention in the graph drawing community; this is particularly due to some attractive properties, namely its analytic nature, which allows to characterize global solutions using spectral decomposition.

In the following, the background of the basic method is presented, in particular results from Euclidean geometry and matrix algebra. Good introductory explanations can be found in Cox and Cox (2001, Section 2.2) and Borg and Groenen (2005, Chapter 12); a useful unified collection of detailed proofs is available in Mardia et al. (1979). Next, the application to graph distances is discussed, before speedup methods are presented.

3.1 Basic Method

The first complete explication was given by Torgerson (1952, 1958). Independently, an equivalent formulation was discovered some years later by Gower (1966), leading to the alternative name *Torgerson-Gower scaling*. As is pointed out, e.g., in Young and Hamer (1987), the problem of constructing coordinates from possibly fallible distance measurements appeared even long before Torgerson's seminal work, e.g. in geodesy and trigonometry (Cayley, 1841; Gauß, 1910; Menger, 1928).

Torgerson's major contribution was the first systematic procedure for computers and the mathematical soundness of the method, especially when dealing with erroneous measurement. For a survey of the history and development of MDS methods, see Mead (1992).

3.1.1 Euclidean Distances

The fundamental starting point in classical scaling is a set $V = \{1, \ldots, n\}$ of objects and the $\binom{n}{2}$ *dissimilarities* among them. The desired output is a set of d-dimensional coordinates such that their Euclidean distance equals the given dissimilarities. In other words, the problem to be solved is:

$$\begin{array}{lll} \text{Given} & D = (d_{ij}) \in \mathbb{R}^{n \times n}, & \\ \text{find} & x_1, \ldots, x_n \in \mathbb{R}^d & \\ \text{such that} & \|x_i - x_j\| = d_{ij} & \text{for all } i, j \in V \end{array} \tag{3.1}$$

Dissimilarity matrices $D = (d_{ij})$ are usually required to have positive entries and to be symmetric, $d_{ij} = d_{ji} \geq 0$, and zero on the main diagonal, $d_{ii} = 0$, for all $i, j \in V$. For the time being, it is assumed that these input dissimilarities are actually Euclidean distances derived from Euclidean coordinates, which are to be reconstructed. In such a case, D is called *Euclidean matrix*.

A procedure to perform this reconstruction is based on results obtained by Schoenberg (1935); Young and Householder (1938). Recall that we are looking for coordinates in d-dimensional space, i.e. a matrix $X \in \mathbb{R}^{n \times d}$ with $X = [x_1, \ldots, x_n]^T$, such that $d_{ij} = \|x_i - x_j\|$. Since distances do not change under translations, we are free to require that the whole configuration be centered in the origin and thus

$$\sum_{i=1}^{n} x_i = 0. \tag{3.2}$$

Squaring on both sides and expanding $\|x_i - x_j\|$ to inner products yields

$$\begin{aligned} d_{ij}^2 &= \|x_i - x_j\|^2 \\ &= \langle x_i - x_j, x_i - x_j \rangle \\ &= \langle x_i, x_i \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle \ , \end{aligned}$$

which, solved for $\langle x_i, x_j \rangle$, gives

$$\langle x_i, x_j \rangle = -\frac{1}{2} \left(d_{ij}^2 - \langle x_i, x_i \rangle - \langle x_j, x_j \rangle \right) \ . \tag{3.3}$$

Using (3.2) and averaging over rows $i = 1, \ldots, n$, over columns $j = 1, \ldots, n$, and over all combinations of i and j, respectively, we obtain

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n} d_{ij}^2 &= \frac{1}{n} \sum_{i=1}^{n} \langle x_i, x_i \rangle + \langle x_j, x_j \rangle \\ \frac{1}{n} \sum_{j=1}^{n} d_{ij}^2 &= \langle x_i, x_i \rangle + \frac{1}{n} \sum_{j=1}^{n} \langle x_j, x_j \rangle \\ \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}^2 &= \frac{2}{n} \sum_{i=1}^{n} \langle x_i, x_i \rangle \ , \end{aligned}$$

which is plugged into (3.3), giving a matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ with entries

$$b_{ij} = \langle x_i, x_j \rangle = -\frac{1}{2} \left(d_{ij}^2 - \frac{1}{n} \sum_{i=1}^{n} d_{ij}^2 - \frac{1}{n} \sum_{j=1}^{n} d_{ij}^2 + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}^2 \right) \tag{3.4}$$

so that the inner products are derived solely from the given distances, still without knowing the original coordinates $x_1, \ldots, x_n$.

This double-centering operation is written in terms of matrices (Schönemann, 1970a, Appendix 1); the structure of this centering operation is studied by Critchley (1988). $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ is a *matrix of inner products*; the double-centering operation in (3.4) reads

$$B = -\frac{1}{2} J D^{(2)} J \ , \tag{3.5}$$

where $D^{(2)}$ denotes matrix D with all entries squared, and $J = I - \frac{1}{n} 1_n 1_n^T$ is a *centering matrix*, with $1_n = [1, \ldots, 1]^T \in \mathbb{R}^n$ and unit matrix $I = \mathrm{diag}(1_n)$, or written in a more expanded form

$$B = -\frac{1}{2} \begin{bmatrix} \frac{n-1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \frac{n-1}{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\frac{1}{n} \\ -\frac{1}{n} & \cdots & -\frac{1}{n} & \frac{n-1}{n} \end{bmatrix} \begin{bmatrix} 0 & d_{12}^2 & \cdots & d_{1n}^2 \\ d_{21}^2 & 0 & \ddots & \vdots \\ \cdots & \ddots & \ddots & d_{n-1,n}^2 \\ d_{n1}^2 & \cdots & d_{n,n-1}^2 & 0 \end{bmatrix} \begin{bmatrix} \frac{n-1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & \frac{n-1}{n} & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\frac{1}{n} \\ -\frac{1}{n} & \cdots & -\frac{1}{n} & \frac{n-1}{n} \end{bmatrix} .$$

Algorithm 1: Classical scaling

Input: dissimilarity matrix $D \in \mathbb{R}^{n \times n}$, dimensionality $d \in \mathbb{N}$
Output: coordinate vectors $x_1, \ldots, x_d \in \mathbb{R}^n$

$B \leftarrow -\frac{1}{2} J D^{(2)} J$
$(\lambda_1, u_1), \ldots, (\lambda_d, u_d) \leftarrow$ decompose B using Algorithm 2
for $i = 1, \ldots, d$ **do**
  $x_i \leftarrow \sqrt{\{\max \lambda_i, 0\}} \cdot u_i$

The coordinate matrix $X \in \mathbb{R}^{n \times d}$ is the desired factor which leads to this matrix of products and for which

$$XX^T = B. \tag{3.6}$$

It is obtained by a spectral decomposition. Since B is symmetric, its eigenvalues and eigenvectors are all real. Let w.l.o.g. $\lambda_1 \geq \cdots \geq \lambda_n \in \mathbb{R}$ and $u_1, \ldots, u_n \in \mathbb{R}^n$ with $\|u_i\| = 1$ and $\langle u_i, u_j \rangle = 0$ denote the eigenvalues and the corresponding unit length eigenvectors. B can be decomposed into

$$B = U \Lambda U^T = \sum_{i=1}^{n} \lambda_i u_i u_i^T \ , \tag{3.7}$$

where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n \times n}$ and $U = [u_1, \ldots, u_n]$.

Since D (and thus B) are derived from a d-dimensional Euclidean space, B has rank d, and thus has d positive eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d > 0$ and $n - d$ zero eigenvalues $\lambda_{d+1} = \cdots = \lambda_n = 0$. Due to the centering operation in (3.5) there is always at least one zero eigenvalue $\lambda_n = 0$.

The $n - d$ zero eigenvalues do not contribute to (3.7) and are ignored; the original d-dimensional coordinates are recovered up to rotation and reflection by setting

$$X = \left[X^{(1)}, \ldots, X^{(d)}\right] = \left[\sqrt{\lambda_1} u_1, \ldots, \sqrt{\lambda_d} u_d\right] = \Lambda^{1/2} U \tag{3.8}$$

in (3.6). Pseudocode for classical scaling is given in Algorithm 1.

3.1.2 Dissimilarities

Up to now it is assumed that D is Euclidean, which allows for perfect reconstruction without errors. In practice, this is not a realistic assumption. For example, geographic distance measurements in two dimensions may result in distance matrices which are slightly erroneous, since errors due to measurement and rounding are often inevitable. The rationale behind the reconstruction process is that in

this example the actual information to be reconstructed is explained by two large eigenvalues, while the error corresponds to positive or negative eigenvalues with small magnitude.

In fact, the spectrum $\lambda_1, \ldots, \lambda_n$ of matrix B is useful for characterizing the *intrinsic dimensionality* of a set of dissimilarities (d_{ij}). Every eigenpair (λ_i, u_i) is associated with a dimension, whose importance is indicated by the sign and the magnitude of λ_i:

- *Large positive:* The dimension significantly contributes to the given dissimilarites and should be considered in the solution.
- *Small positive:* The dimensions contribute only marginally, and may be ignored.
- *Small negative:* Intuitively, these components make the dissimilarities "slightly non-Euclidean", and the corresponding dimensions may be ignored.
- *Large negative:* Classical scaling may not be appropriate for the analysis of the given dissimilarity.

In practice, the dimensions corresponding to negative eigenvalues are ignored by using only dimensions corresponding to positive eigenvalues and completing the rest with zeroes through

$$\begin{aligned} \lambda_1^+ = \max\{\lambda_1, 0\}, \ldots, \lambda_n^+ = \max\{\lambda_n, 0\} \\ \Lambda^+ = \operatorname{diag}\left(\lambda_1^+, \ldots, \lambda_n^+\right) \end{aligned} \tag{3.9}$$

and thus a modified decomposition

$$B^+ = U\Lambda^+ U^T = \sum_{i=1}^{n} \lambda_i^+ u_i u_i^T \ . \tag{3.10}$$

3.1.3 Strain

Using the matrix approximation results in Eckart and Young (1936), it can be shown that the solution in (3.8) minimizes the loss function *strain*

$$\begin{aligned} \operatorname{strain}(X, D) &= \left\| -\tfrac{1}{2} J D^{(2)} J - X X^T \right\|^2 \\ &= \left\| B - X X^T \right\|^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} (b_{ij} - \langle x_i, x_j \rangle)^2 \end{aligned} \tag{3.11}$$

among all d-dimensional configurations. The value zero is attained if D is Euclidean of dimension d. The term "strain" was introduced by Carroll and Chang (1972); a proof for optimality is given in Mardia (1978); see also Gower (1966).

The robustness of classical scaling and its sensitivity to measurement errors is discussed by Sibson (1978). Further perturbation analysis based on derivatives of the strain criterion is done by Lewis and Trosset (2006); see also Krzanowski (2006).

To assess the "goodness of fit" of the resulting d-dimensional configuration, the spectrum of B can be used. Specifically, Mardia (1978) proposes several criteria to express the intrinsic structure captured almost entirely by the highest d eigenvalues, relative to the remaining information explained by eigenvalues not used. The suggested criteria are

$$p_1(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n-1} \lambda_i} \tag{3.12}$$

in the case where no negative eigenvalues are present and

$$p_2(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} |\lambda_i|} \tag{3.13}$$

or

$$p_3(d) = \frac{\sum_{i=1}^{d} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2} \tag{3.14}$$

otherwise. When d is not given in advance, these criteria are also useful for determining a suitable number of output dimensions, as recommended by Sibson (1978).

3.1.4 Computation

The numerical computation of the decomposition is usually done iteratively. Since in most applications $d = 2$ or $d = 3$, and only d largest positive eigenvalues and associated the eigenvalues are required, a simple *power iteration* is sufficient (Hotelling, 1933; Wilkinson, 1965). An initial vector is iteratively multiplied with B. It can be shown (Hotelling, 1933; Wilkinson, 1965) that the series Bq, B^2q, B^3q converges in direction to the eigenvector u_1 corresponding to the eigenvalue λ_1 of B with the largest magnitude for real symmetric matrices B, i.e.,

$$\lim_{t\to\infty} \frac{B^t q}{\|B^t q\|} = u_1, \tag{3.15}$$

provided that the initial guess q is not the zero vector, or orthogonal to another eigenvector. This series is simply computed by carrying out multiplications

$$q^{[t+1]} \leftarrow \frac{Bq^{[t]}}{\|Bq^{[t]}\|}, \tag{3.16}$$

where normalization of the current iterate controls the growth of $q^{[t]}$. The initial guess $q^{[0]}$ may be chosen randomly; if there is a configuration available, e.g. before some small changes to the original dissimilarities, the iteration may be initialized accordingly, which tends to reduce the number of iteration steps and reflection and rotation artifacts.

The step in (3.16) is carried out until the current iterate remains essentially invariant under multiplication with B, i.e.

$$\left\langle q^{[t+1]}, q^{[t]} \right\rangle > 1 - \epsilon. \tag{3.17}$$

for some accuracy parameter $\epsilon > 0$, assuming that $q^{[t]}, q^{[t+1]}$ are normalized to unit length. The estimate of the corresponding largest eigenvalue is given by

$$\lambda^{[t]} = \left(q^{[t]}\right)^T Bq^{[t]}, \tag{3.18}$$

or, after sufficient convergence, by

$$\lambda^{[t+1]} = \|Bq^{[t]}\|. \tag{3.19}$$

The smaller eigenvalues and corresponding eigenvectors are obtained in a similar fashion, using the fact that the d-th eigenvector u_d of B is the largest eigenvector of matrix B in which the contributions of the $d-1$ previous eigenvalues in (3.7) are removed,

$$\sum_{i=d}^{n} \lambda_i u_i u_i^T = B - \sum_{i=1}^{d-1} \lambda_i u_i u_i^T \tag{3.20}$$

so that the eigenvector u_d is computed in a power iteration by orthogonalizing against the $d-1$ previous eigenvectors with

$$u_d^{[t+1]} = \frac{Bu_d^{[t]}}{\|Bu_d^{[t]}\|} - \sum_{i=1}^{d-1} \frac{\langle u_i^{[t]}, u_d^{[t]} \rangle}{\langle u_i^{[t]}, u_i^{[t]} \rangle} u_i \ , \tag{3.21}$$

where $u_d^{[t]}$ denotes the iterate of u_d after t steps. The power method is summarized in Algorithm 2. Note that the eigenvalues thus obtained are decreasing in *absolute* value, i.e. $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$. It may thus be necessary to

Algorithm 2: Power iteration for spectral decomposition

Input: symmetric matrix $M \in \mathbb{R}^{n \times n}$, dimensionality $d \in \mathbb{N}$
Output: eigenvalues $\lambda_1, \ldots, \lambda_d \in \mathbb{R}$, eigenvectors $u_1, \ldots, u_d \in \mathbb{R}^n$

$u_1, \ldots, u_d \leftarrow$ random
while *(relative change in any u_i)* $> \epsilon$ **do**
  for $i = 1, \ldots, d$ **do**
    $u_i \leftarrow \frac{Mu_i}{\|Mu_i\|}$
    for $j = 1, \ldots, i-1$ **do**
      $u_i \leftarrow u_i - \frac{\langle u_i, u_j \rangle}{\langle u_j, u_j \rangle} u_j$
for $i = 1, \ldots, d$ **do**
  $\lambda_i \leftarrow u_i^T M u_i$

actually compute eigenvectors $u_{d+1}, u_{d+2}, \ldots$ until the d-th *positive* eigenvalue is found, but this rarely occurs in practice when $d \in \{2, 3\}$.

The eigenvalues of $B + c \cdot I$ are $\lambda_1 + c, \ldots, \lambda_n + c$ for any constant $c \in \mathbb{R}$ while the eigenvectors remain unchanged, since for an eigenpair (λ, x)

$$(B + cI) \cdot x = Bx + cIx = \lambda x + cx = (\lambda + c)x.$$

The positive eigenvalues of B are computed by setting c to be at least the absolute value of the largest negative eigenvalue, thus shifting the spectrum by c and making all eigenvalues of $B + c \cdot I$ positive.

Convergence of this method depends linearly on the ratio $|\lambda_2|/|\lambda_1|$, which is occasionally called *eigengap*. If this ratio is large, i.e., if the desired eigenvector is separated well enough from its successor with respect to magnitude, the convergence will be fast; otherwise, it may be improved by precomputing powers B^2, B^4, B^8, etc., and using them in (3.16) instead of B. The running time is in $\mathcal{O}(n^3)$, since the power iteration requires $\mathcal{O}(n)$ steps to converge, each of which encompasses $\mathcal{O}(n^2)$ individual operations; in practice, it can be assumed that a constant number of steps is sufficient, leading to $\mathcal{O}(n^2)$ running time.

The power method is accurate, efficient, and simple enough for practical problems in which

- the matrix to be decomposed is symmetric and real
- only extremal eigenvalues and eigenvectors are desired,
- all of the extremal eigenvalues are positive, and
- the positive extremal eigenvalues are well enough separated.

If these assumptions cannot be made, other methods may be used. More detailed treatments of power iteration and more sophisticated methods for the numerical computation of spectral decompositions are given by Golub and van Loan (1996).

3.2 Classical Scaling for Graphs

Multidimensional scaling seems to have been the first computerized layout method used for drawing social networks towards the end of the 1960s. At that time, the stress minimization approach had just established in other disciplines like statistics and psychometrics, while the classical variant was used less frequently. Therefore, there is no early explicit reference for the application of classical scaling to graph drawing, even though spectral graph drawing methods were known at that time, most notably by Hall (1970); see also Koren (2003) and Section 4.2.

Probably among the first to use classical scaling and other MDS variants for assigning positions to objects in a network were Kruskal and Seery (1980). The MDS solution is used mainly to guide a "network designer" to find a useful initial overall configuration, which may then be manually rearranged.

In the graph drawing and information visualization literature, classical scaling was rediscovered only much later, within interactive high dimensional layout (Hosobe, 2004, 2005) and comparative studies (Buja et al., 2001; Koren and Harel, 2005), before the method itself was in the focus of interest in (Brandes and Pich, 2007; Çivril et al., 2006, 2007). Considerable attention was raised by an application within a method for *nonlinear dimensionality reduction* published in a *Science* article by Tenenbaum et al. (2000).

3.2.1 Graph Distances

In graph drawing and network analysis, "distance" most frequently refers to shortest-path distances, which are the natural choice for the dissimilarity D to be used as input in MDS. Two nodes are considered most proximate when they are directly joined by an edge and have distance 1. The proximity of non-adjacent nodes is then inversely proportional to the distance, i.e., the minimum number of steps that have to be made via adjacent nodes.

An alternative interpretation of the distance matrix D thus computed is that of an adjacency matrix of the induced *complete graph* in which all possible edges $\{u, v\}$ $(u, v \in V)$ are present, with weights determined by the shortest-path distance d_{uv}. When the edge lengths, and hence the magnitudes of node proximities, are not uniform, the minimum sum of edge lenghts along a path is

considered instead. Note that two adjacent nodes u, v may be more proximate than what is given by the weight $w(u, v)$ of the corresponding incident edge, since there may be a detour via some other nodes, which has a higher number of edges, but a lower sum of edge weights.

Computing the matrix D of shortest-path distances is a fundamental algorithmic problem. If the entire matrix is needed, the *all-pairs shortest-paths problem* (APSP) needs to be solved, using one of the standard algorithms (let $[D]_{ij}$ denote the ij entry of matrix D):

- *Floyd-Warshall:* Using a method reminiscent of matrix multiplication (Floyd, 1962; Warshall, 1962), the distance matrix D is computed by starting with a preliminary distance matrix

$$\left[D^{[0]}\right]_{ij} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} , \end{cases} \tag{3.22}$$

  and computing the matrices $D^{[t]}$ for $t > 0$, which contain lengths of shortest paths of at most t edges, by relaxing

$$\left[D^{[t+1]}\right]_{ij} = \min_{k \in V} \left\{ \left[D^{[t]}\right]_{ik} + \left[D^{[t]}\right]_{kj} \right\} \tag{3.23}$$

  for each entry ij until the eventual distance matrix $D = D^{[\mathrm{diam}(G)]}$ has been computed. The resulting running time is in $\mathcal{O}(n^2 \cdot \mathrm{diam}(G))$.

- *Bellman-Ford:* Sparsity of the graph is exploited by replacing the relaxation step in (3.23) by

$$\left[D^{[t+1]}\right]_{ij} = \min_{k \in N(j)} \left\{ \left[D^{[t]}\right]_{ik} + w(k, j) \right\} \ ; \tag{3.24}$$

  here, $D^{[n]}$ has to be computed, giving an overall running time of $\mathcal{O}(nm)$ (Bellman, 1958; Ford, 1956).

- *Breadth-First Search, Dijkstra:* If only a subset of the columns of matrix D is needed, it is more advantageous to employ an algorithm for solving the *single-source shortest paths* problem (SSSP) and carrying it out starting from all the corresponding source nodes. If edges are uniformly weighted, given a single source node $v \in V$, an elementary algorithm such as breadth-first search computes the distances $d_{vw}, w \in V$, in $\mathcal{O}(m)$ time. For general nonnegative edge weights, the algorithm of Dijkstra (1959) requires $\mathcal{O}(m + n \log n)$ time if implemented with Fibonacci heaps (Fredman and Tarjan, 1987). For a comprehensive survey of elementary graph traversal algorithms, see Cormen et al. (2001).

A consequence from the optimality criteria leading to (3.23) and (3.24) is that shortest-path distances are *metric*, i.e, they satisfy

$$\begin{array}{rcll} d_{uu} & = & 0 & \text{(reflexivity)} \\ d_{uv} & = & d_{vu} & \text{(symmetry)} \\ d_{uw} & \leq & d_{uv} + d_{vw} & \text{(triangle inequality)} \end{array}$$

for all $u, v, w \in V$, so that (V, d) constitutes a *metric space* with the metric function $d\colon V \times V \to \mathbb{R}_{\geq 0}$, where $d(u, v) \equiv d_{uv}$. These metric properties of shortest-path distances are crucial for the application of classical scaling and justify that D is used as the input to the MDS procedure directly.

Buckley and Harary (1989) give a comprehensive study of distances in graphs and the relation to other graph characteristics from a graph-theoretical point of view. A survey of related and more general network statistics is given, e.g., by Brinkmeier and Schank (2005).

A characteristic feature of classical scaling is that the contribution of a dissimilarity to the objective (3.11) is weighted proportionally to its squared value. In effect, the fit of larger dissimilarities is favored over the fit of smaller ones. Objective functions and their weights are discussed in more detail in Chapter 4; the experimental study in Chapter 5 discusses the practical impact of implicit and explicit weights.

3.2.2 Eigenvalues

When B is derived from shortest-path distances in a given graph, the spectrum of the derived matrix B indicates how far away its shortest paths are from being Euclidean, and how useful classical scaling actually is for analyzing this type of distance.

Figure 3.1 shows a matrix of layouts for an example graph in the five most significant dimensions of matrix B. The solutions obtained when minimizing the objective function in (3.11) are nested dimension-wise. For example, the strain-optimal three-dimensional configuration for a given D contains the strain-optimal two-dimensional solution. Abelson and Messick (1956) propose to compute the solution dimension after dimension and to evaluate one of the criteria (3.12)–(3.14) and to use the identities

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = n\sum_{i=1}^{n}\langle x_i, x_i\rangle = n\,\mathrm{tr}B = n\sum_{i=1}^{n-1}\lambda_i \;, \tag{3.25}$$

to determine the required sum of eigenvalues in the denominators.

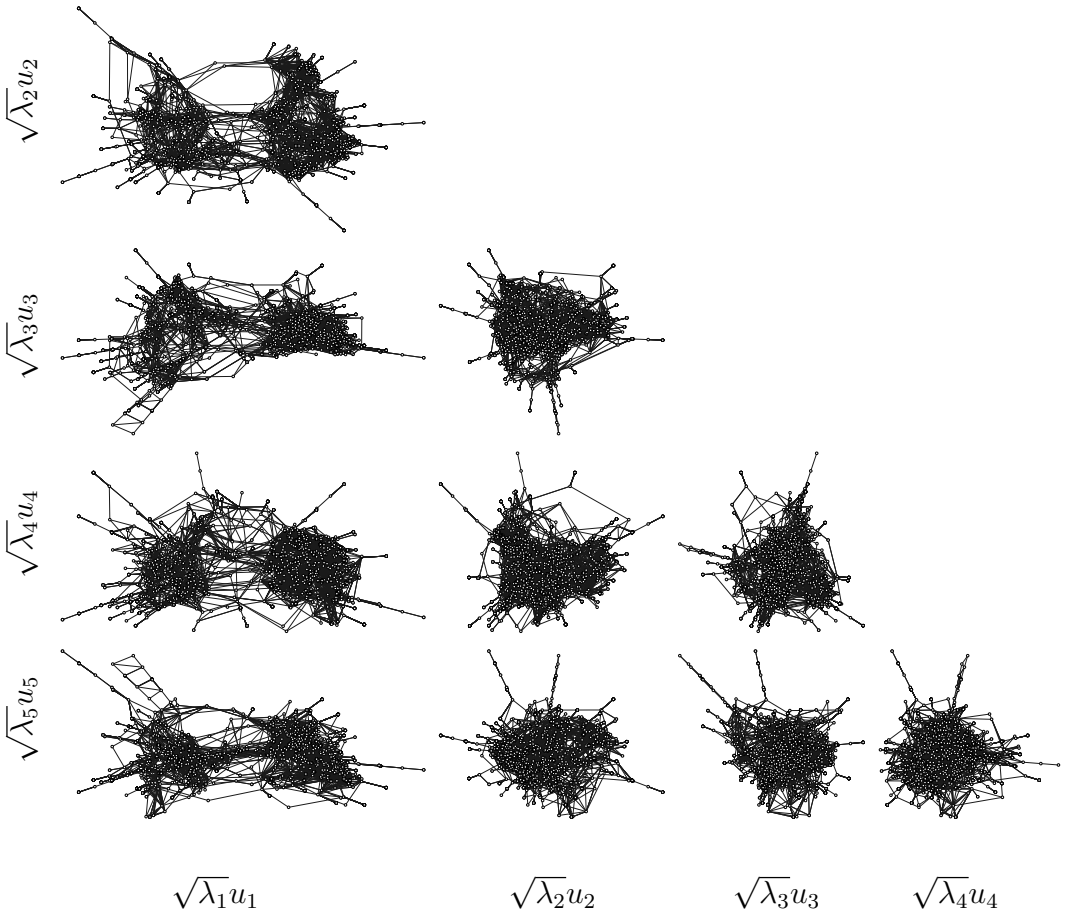


Figure 3.1: Layout matrix of all possible combinations of the five dimensions associated with the largest eigenvalues $\lambda_1 \approx 11192.09, \lambda_2 \approx 4227.92, \lambda_3 \approx 2881.86, \lambda_4 \approx 2717.65, \lambda_5 \approx 2365.57$ of matrix B for the esslingen1 graph (2075 nodes, 4769 edges).

The extremal regions of the spectra of ten example graphs are shown in Figure 3.2. A striking observation is that 516 and plat1919 have no negative eigenvalues and only two or three positive eigenvalues; in both of them there is a close correspondence to original two- or three-dimensional coordinates, which classical scaling is able to reconstruct. Despite some small negative eigenvalues, the steep decrease in the positive eigenvalues of 1138bus, esslingen1, and sw002 indicates that using the largest positive eigenvalues and their eigenvalues captures most of the distance information. In contrast, the presence of larger negative eigenvalues and the slow decrease of the positive ones in sw01 and prot1 suggest that any choice of dimensions would reflect only a small part of the original distances. These graphs are also used in the experimental study in Chapter 5.

Other notions of low-dimensional graph representation are also studied, using an alternative definition of dimension (Harary and Melter, 1976). Indyk (2001); Linial et al. (1995) discuss embeddings with low distortion; realizability and rigidity of graph distances are studied by Alfakih (2000), using results from Euclidean geometry very closely related to classical scaling. The spectral properties of graph distance matrices are frequently studied in mathematical chemistry in the reconstruction of molecular conformations (Mihalić et al., 1992; Randić et al., 1994).

3.3 Speedup

Running time and space complexity of classical scaling are at least quadratic in the number of objects, since even only storing all dissimilarities in an $n \times n$ matrix already requires n^2 operations. When implemented using the power iteration scheme outlined in Section 3.1.4, the running time per iteration is in $\Theta(n^2)$. In the context of graph drawing, this is not a problem for small and medium graphs of up to several thousands of nodes, but makes classical scaling impractical for larger graphs.

Poor scalability to large data sets due to quadratic complexity is a well-known problem of all MDS algorithms. It was addressed as early as in the 1960s (Kruskal and Hart, 1966), and since then, many approaches to speeding up the computation of coordinates have been devised (Chalmers, 1996; Jourdan and Melançon, 2004; Morrison and Chalmers, 2003; Morrison et al., 2002). Likewise, methods for speeding up layout methods using the decomposition of other matrices, usually the Laplacian, have been proposed (Faloutsos and Lin, 1995; Wang et al., 1999). Relationships between these approaches are discussed by Bengio et al. (2004) and Platt (2004). For more general surveys on sparse techniques for dimensionality reduction and related spectral methods, see Burges (2004) and (Saul et al., 2005).

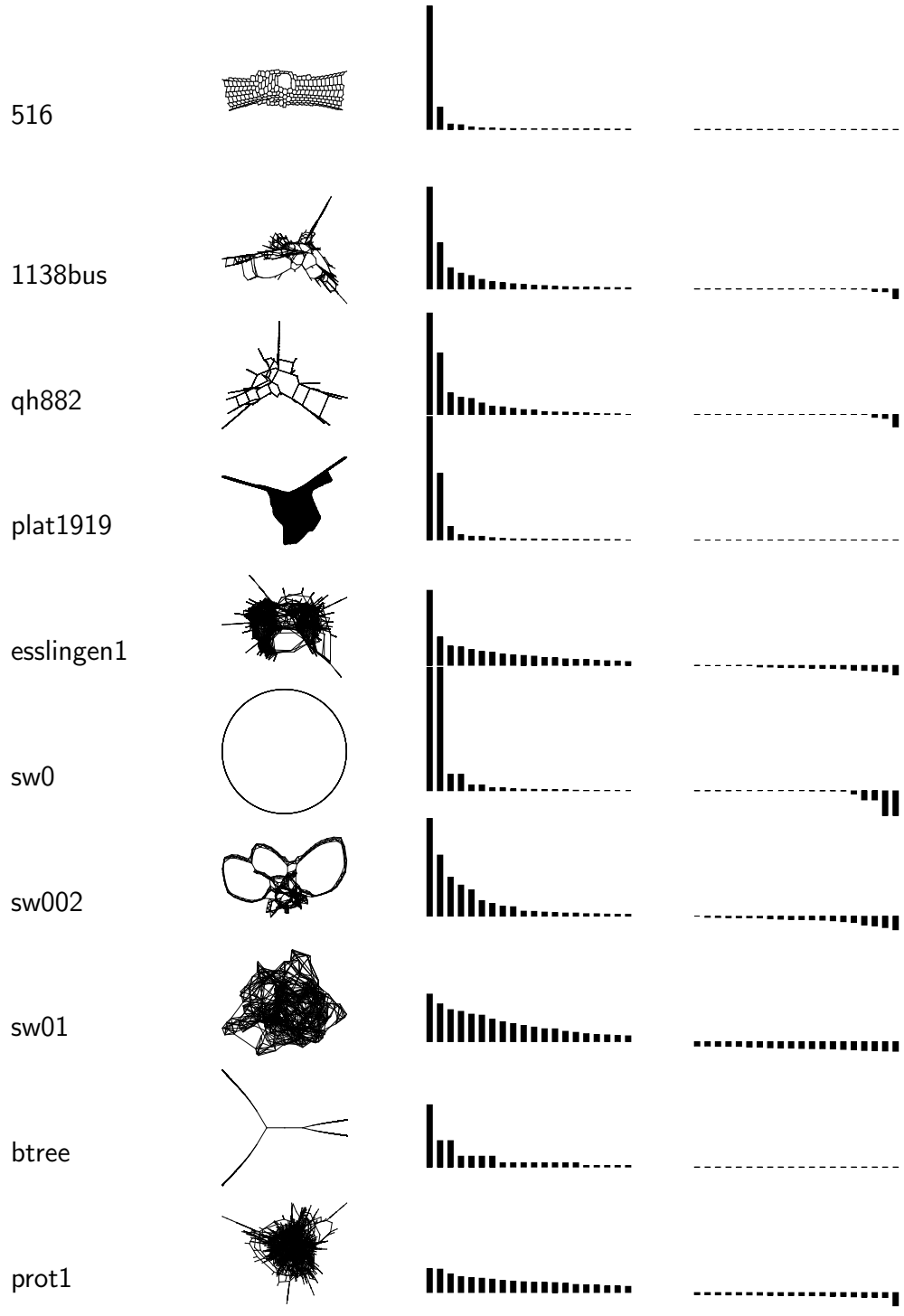


Figure 3.2: Extremal eigenvalues of the matrix B for some example graphs, normalized by $\operatorname{tr} B$. The layout uses the eigenvectors of the two largest positive eigenvalues.

3.3.1 Pivoting

Instead of computing all n columns of B, a rectangular matrix $C \in \mathbb{R}^{n \times k}$ is constructed by picking a small number $k \ll n$ of *pivot* nodes from a graph $G = (V, E)$, which should be representative and capture as much of the distance information as possible. Each of the k columns of C contains the shortest-path distances from each pivot to all other nodes. Note that, alternatively, the term *landmark* will be used later. While pivot nodes and landmark nodes are identical in practice, their names correspond to distinct MDS computation procedures, which is reflected in distinct names.

Various pivoting strategies are available (Silva et al., 2005). For each pivot $p_i \in P$, the distances $d(p_i, v), v \in V$ are represented by a column in the corresponding *pivot matrix* D_P

$$\begin{bmatrix} d_{v_1 p_1} & \cdots & d_{v_1 p_k} \\ d_{v_2 p_1} & \cdots & d_{v_2 p_k} \\ d_{v_3 p_1} & \cdots & d_{v_3 p_k} \\ \vdots & & \vdots \\ d_{v_n p_1} & \cdots & d_{v_n p_k} \end{bmatrix} \in \mathbb{R}^{n \times k}, \tag{3.26}$$

so that the k columns of D_P are a subset of the n columns of D.

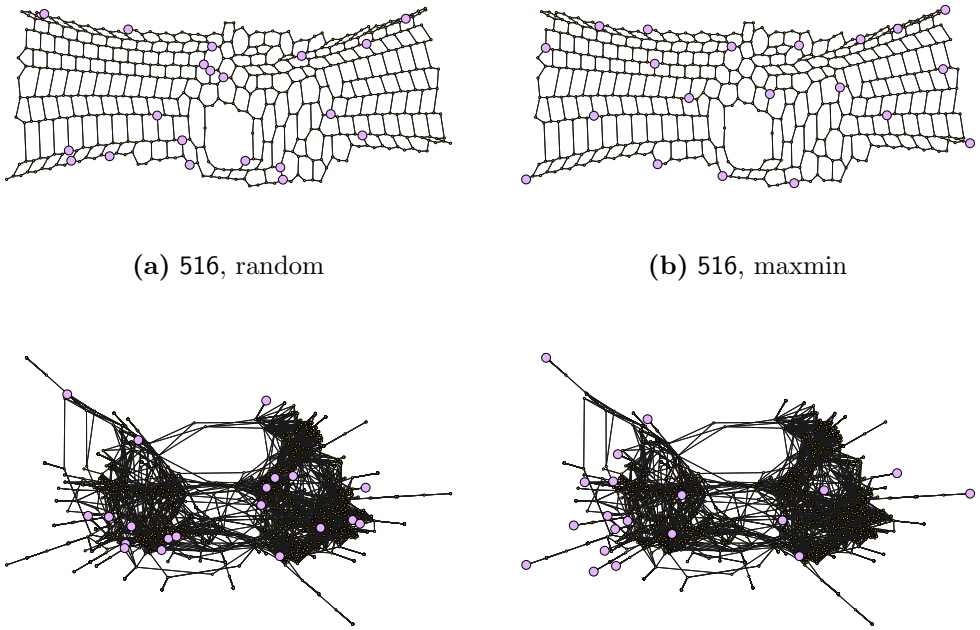
- *Random sampling:* Nodes are sampled from V with uniform probability. Sampling may be done with or without replacement. Alternatively, probabilities may be proportional, e.g., to node degree.

- *Maxmin:* The first pivot node p_1 is picked at random. Then, for $i \in \{2, \ldots, k\}$, the i-th pivot is picked to be farthest away from any of the $i-1$ pivots selected so far by

$$p_i \leftarrow \underset{v \in V \setminus \{p_1, \ldots, p_{i-1}\}}{\operatorname{argmax}} \min_{p \in \{p_1, \ldots, p_{i-1}\}} d(v, p). \tag{3.27}$$

  Pseudo-code is given in Algorithm 3. Remote parts of the graph, such as end nodes of attached paths, are likely to be populated with pivots, thereby achieving a good scattering of pivots over the graph, which is thus seen from many different viewpoints in terms of the distances. This is a well-known 2-approximation to the $\mathcal{NP}$-complete k-centers problem in facility location (Hochbaum and Shmoys, 1985).

- *Hybrid strategy:* Like maxmin, but in between, every j-th pivot is picked at random so as to lessen systematic errors which may occur because too many nodes at the end of attached paths are selected as pivots.



(a) 516, random **(b)** 516, maxmin

(c) esslingen1, random **(d)** esslingen1, maxmin

Figure 3.3: Picking 20 pivots randomly and with the maxmin strategy, in the graphs 516 and esslingen1. The layout is done with full classical scaling. Maxmin scatters nodes more systematically, but also tends to overpopulate peripheral regions with pivots.

3.3.2 Pivot MDS

Pivot MDS reduces the quadratic time and space complexity of the unmodified classical scaling approach. It is based on a decomposition of the rectangular matrix D_P in (3.26) corresponding to a pivot set $P \subseteq V$. A layout is obtained using the singular vectors of a rectangular matrix, instead of the eigenvectors of a square matrix. If $k \ll n$, the running time of classial scaling is reduced significantly.

Let d_{vp} be the shortest-path distance between node $v \in V$ and pivot $p \in P$. Analogously to (3.2), coordinates x_v are sought for all nodes $v \in V$ such that $d_{vp} \approx \|x_v - x_p\|$, or expressed as a squared inner product,

$$d_{vp}^2 \approx \langle x_v - x_p, x_v - x_p \rangle = \langle x_v, x_v \rangle + \langle x_p, x_p \rangle - \langle 2x_v, x_p \rangle \tag{3.28}$$

and thus

$$\langle x_v, x_p \rangle \approx -\frac{1}{2} \left(d_{vp}^2 - \langle x_v, x_v \rangle - \langle x_p, x_p \rangle \right). \tag{3.29}$$

Algorithm 3: Construction of a pivot set

Input: undirected connected graph $G = (V, E)$, number $k \in \mathbb{N}$ of pivots
Output: pivot set $P = \{p_1, \dots, p_k\} \subseteq V$, distance matrix $D_P \in \mathbb{R}^{n \times k}$

$p_1 \leftarrow$ random $\in V$
for $v \in V$ **do**
  $m_v \leftarrow \infty$
for $i = 1, \dots, k$ **do**
  ith column of $D_P \leftarrow [d_{p_i v_1}, \dots, d_{p_i v_n}]^T$
  for $v \in V$ **do**
    $m_v \leftarrow \min\{m_v, d_{p_i v}\}$
  $p_{i+1} \leftarrow \operatorname{argmax}_{v \in V} m_v$

Summing over all nodes $v \in V$ in (3.28) and dividing by n gives

$$\begin{aligned} \frac{1}{n} \sum_{v \in V} d_{vp}^2 &= \frac{1}{n} \sum_{v \in V} \langle x_v, x_v \rangle + \frac{1}{n} \sum_{v \in V} \langle x_p, x_p \rangle - \frac{2}{n} \left\langle \underbrace{\sum_{v \in V} x_v}_{=0}, x_p \right\rangle \\ &= \frac{1}{n} \sum_{v \in V} \langle x_v, x_v \rangle + \langle x_p, x_p \rangle, \end{aligned} \tag{3.30}$$

where we assume that the configuration of all nodes is centered in the origin, as required in (3.2). In addition, we assume that the k pivots are well spread over the graph, so that the configuration of only these k nodes also has its barycenter in the origin, i.e.

$$\sum_{p \in P} x_p = 0; \tag{3.31}$$

by averaging over all pivots $p \in P$ in (3.28), we get

$$\begin{aligned} \frac{1}{k} \sum_{p \in P} d_{vp}^2 &= \frac{1}{k} \sum_{p \in P} \langle x_v, x_v \rangle + \frac{1}{k} \sum_{p \in P} \langle x_p, x_p \rangle - \frac{2}{k} \left\langle x_v, \underbrace{\sum_{p \in P} x_p}_{=0} \right\rangle \\ &= \langle x_v, x_v \rangle + \frac{1}{k} \sum_{p \in P} \langle x_p, x_p \rangle, \end{aligned} \tag{3.32}$$

and, likewise, in (3.30),

$$\begin{aligned}\frac{1}{nk}\sum_{v\in V}\sum_{p\in P} d_{vp}^2 &= \frac{1}{nk}\sum_{v\in V}\sum_{p\in P}\langle x_v, x_v\rangle + \frac{1}{k}\sum_{p\in P}\langle x_p, x_p\rangle \\ &= \frac{1}{n}\sum_{v\in V}\langle x_v, x_v\rangle + \frac{1}{k}\sum_{p\in P}\langle x_p, x_p\rangle.\end{aligned} \tag{3.33}$$

Rearranging (3.30) and (3.32) gives

$$\langle x_v, x_v\rangle = \frac{1}{k}\sum_{p\in P} d_{vp}^2 - \frac{1}{k}\sum_{p\in P}\langle x_p, x_p\rangle \tag{3.34}$$

and

$$\langle x_p, x_p\rangle = \frac{1}{n}\sum_{v\in V} d_{vp}^2 - \frac{1}{n}\sum_{v\in V}\langle x_v, x_v\rangle. \tag{3.35}$$

Plugging (3.34) and (3.35) into (3.29) and using (3.33) eventually yields

$$\langle x_v, x_p\rangle = -\frac{1}{2}\left(d_{vp}^2 - \frac{1}{k}\sum_{p\in P} d_{vp}^2 - \frac{1}{n}\sum_{v\in V} d_{vp}^2 + \frac{1}{nk}\sum_{v\in V}\sum_{p\in P} d_{vp}^2\right). \tag{3.36}$$

Similarly to the matrix B in (3.5), the *inner product* matrix $C \in \mathbb{R}^{n\times k} = (c_{v_i p_j})$ is obtained by setting c_{vp} to be the right hand side of (3.36). Again, this is written more compactly as

$$C = -\frac{1}{2} J_n D_P^{(2)} J_k, \tag{3.37}$$

where $J_n \in \mathbb{R}^{n\times n}$, $J_k \in \mathbb{R}^{k\times k}$ are centering matrices defined analogously to the one used in (3.5), with side lengths n and k, respectively.

Without having to compute B, the desired eigenvectors of B are approximated using the smaller matrix C. It is exploited that the eigenvectors of $BB^T = B^2$ are identical to those of B, but with all eigenvalues squared. The eigenvectors of BB^T are estimated using $CC^T \in \mathbb{R}$. If $V = P$, Pivot MDS is equivalent to decomposing the matrix B^2.

An intuitive interpretation is that the eigenvectors of CC^T approximate the eigenvectors of $BB^T = B^2$, and thus of B. When $[A]_{uv}$ denotes the entry of matrix A for the nodes $u, v \in V$, this follows from the assumption

$$\left[B^2\right]_{uw} = \left[BB^T\right]_{uw} = \sum_{v\in V} b_{uv} b_{wv} \approx \sum_{p\in P} c_{up} c_{wp} = \left[CC^T\right]_{uw}, \tag{3.38}$$

so matrix entries $[B^2]_{uw}$ and $[CC^T]_{uw}$ represent the same type of transformed distance sums, though in the latter case with a truncated list of intermediaries.

Algorithm 4: Pivot MDS

Input: undirected connected graph $G = (V, E)$, dimensionality $d \in \mathbb{N}$
Output: coordinate vectors $x_1, \ldots, x_d \in \mathbb{R}^n$

$D_P \leftarrow$ pivot matrix for G generated with Algorithm 3
$C \leftarrow -\frac{1}{2} J_n D_P^{(2)} J_k$
for $i = 1, \ldots, d$ **do**
  $y_i \leftarrow$ random $\in \mathbb{R}^k$
$H \leftarrow C^T C$
while *(relative change in any* y_i*)* $> \epsilon$ **do**
  for $i = 1, \ldots, d$ **do**
    $y_i \leftarrow \frac{H y_i}{\|H y_i\|}$
    for $j = 1, \ldots, i-1$ **do**
      $y_i \leftarrow y_i - \frac{\langle y_i, y_j \rangle}{\langle y_j, y_j \rangle} y_j$
for $i = 1, \ldots, d$ **do**
  $x_i \leftarrow C y_i$

If these are sufficiently well distributed, the relative size of entries in CC^T is representative for those in B^2.

To compute the first eigenvector of CC^T, a power iteration is carried out by starting with some non-zero initial vector $y \in \mathbb{R}^n$ and computing

$$\lim_{i \to \infty} \left(CC^T\right)^i y. \tag{3.39}$$

by iteratively computing

$$y \leftarrow \frac{CC^T y}{\|CC^T y\|}. \tag{3.40}$$

Using associativity of matrix multiplication, this can be rearranged to

$$\lim_{i \to \infty} \underbrace{CC^T \cdots CC^T}_{i} x = \lim_{i \to \infty} C \underbrace{C^T C \cdots C^T C}_{i-1} C^T y = C \left(\lim_{i \to \infty} \left(C^T C\right)^i \right) C^T y.$$

The power iteration consists of two steps: first, positions of pivots are determined using the current positions of all items (multiplication with $C^T \in \mathbb{R}^{k \times n}$), and then all items are positioned relatively to the pivots (multiplication with $C \in \mathbb{R}^{n \times k}$). Note that it is this interpretation that motivates the name "pivot," in contrast to "landmarks" which are first assigned their final location and then used to determine the position of all other items.

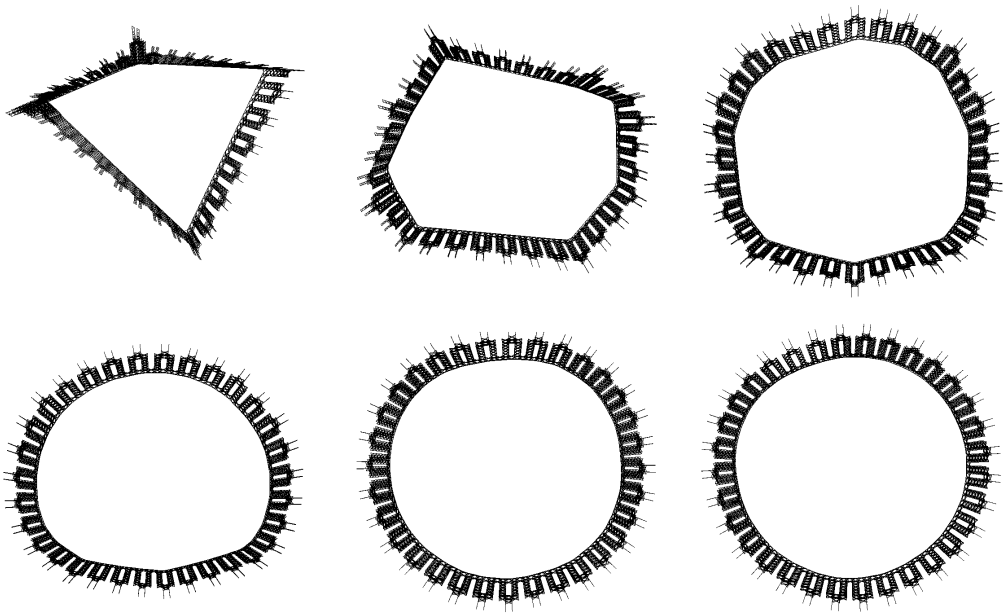


Figure 3.4: Progressively drawing the finan512 graph (74 752 nodes 261 120 edges) with increasing pivot set ($k = 3, 6, 12, 25, 50, 100$) using pivoting with the maxmin strategy.

With a fixed k and assuming a constant number of iteration steps, the running time of computing the singular value decomposition of C is in $\mathcal{O}(k^2 n)$ instead of $\mathcal{O}(n^2)$.

The power iteration process in (3.40) is justified by the singular value decomposition of C. Let $u_1, \ldots, u_k \in \mathbb{R}^n$ be the left singular vectors, $v_1, \ldots, v_k \in \mathbb{R}^k$ be the right singular vectors, and $\sigma_1, \ldots, \sigma_k \in \mathbb{R}$ be the singular values of C. Without loss of generality, $\|u_i\| = \|v_i\| = 1$ for all $i \in \{1, \ldots, k\}$. Then, the ith left and right singular vectors and the ith singular value are related by

$$Cu_i = \sigma_i v_i \quad \text{and} \quad C^T v_i = \sigma_i u_i \tag{3.41}$$

and thus

$$CC^T u_i = \sigma_i^2 u_i \quad \text{and} \quad C^T C v_i = \sigma_i^2 v_i, \tag{3.42}$$

which implies that $u_1, \ldots, u_k$ and $\sigma_1^2, \ldots, \sigma_k^2$ are eigenvectors and eigenvalues of $CC^T \in \mathbb{R}^{n \times n}$ (which is of rank k), and $v_1, \ldots, v_k$ and $\sigma_1^2, \ldots, \sigma_k^2$ are eigenvectors and eigenvalues of $C^T C \in \mathbb{R}^{k \times k}$

It is natural to trade off running time and quality of the estimate. To find a suitable value for the number k of pivots, the user may have to experiment with various values of k, construct corresponding pivot sets of size k, and re-execute

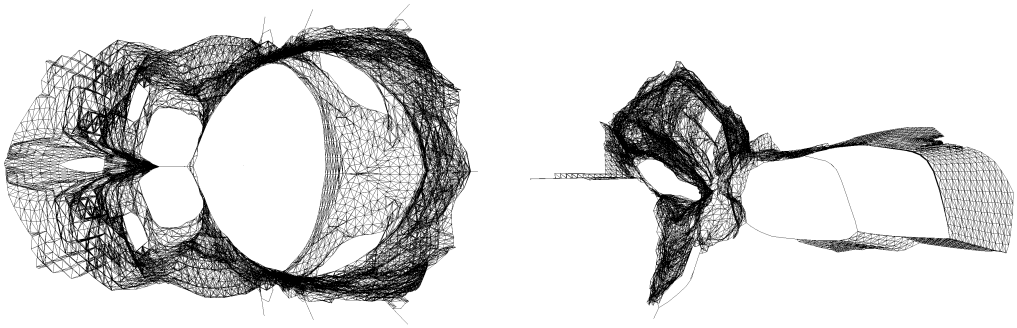


Figure 3.5: Pivot MDS drawings of the graphs bcsstk31 (35 588 nodes, 572 914 edges) and bcsstk32 (44 609 nodes, 985 046 edges) with 200 pivots. In the experimental study of Hachul and Jünger (2006) these graphs posed serious difficulties for most methods.

the Pivot MDS procedure. Alternatively, the solutions can be constructed progressively using previous results and incrementing the pivot set until the output is satisfactory or stable.

This may be desirable for drawing very large graphs. An extremely quick initial layout is obtained with only a few pivots; while this layout is already presented graphically, it is further refined in the background by incrementing the set of pivots, as in Figure 3.4. In fact, it is often observed in experiments that the majority of running time is consumed by distance computations, while computing layouts from these distances is fast; this is due to the advanced data structures that are necessary for graph traversal, while the classical scaling layout require only elementary matrix operations. Therefore, it is desirable to execute as few instances of the SSSP procedure as possible. The asymptotic running time complexities are summarized in Table 3.1. See Chapter 5 for an experimental comparison.

3.3.3 Landmark MDS

Another speedup technique goes back to de Silva and Tenenbaum (2003) and is called Landmark MDS. It is very similar to Pivot MDS in that it also operates on a subset of the complete distance information; the pivots, here called *landmarks*, may be picked from all nodes using one of the strategies discussed in Section 3.3.1. However, these two approaches process this distance information in different ways.

Let $L = \{l_1, \ldots, l_k\} \subseteq V$ be a set of landmarks. The first step is classical

scaling restricted to the k landmarks, with the distance matrix

$$D_L = \begin{bmatrix} 0 & d_{l_1 l_2} & \dots & d_{l_1 l_k} \\ d_{l_2 l_1} & 0 & \dots & d_{l_2 l_k} \\ \vdots & \vdots & \ddots & \vdots \\ d_{l_k l_1} & d_{l_k l_2} & \dots & 0 \end{bmatrix} \in \mathbb{R}^{k \times k}, \tag{3.43}$$

and the derived matrix

$$B_L = -\frac{1}{2} J_k D_L^{(2)} J_k, \tag{3.44}$$

which is decomposed into $B_L = U \Lambda U^T$, where $U, \Lambda \in \mathbb{R}^{k \times k}$, similar to (3.7), and J_k is the centering matrix as in (3.4). Let $\lambda_1 \geq \dots \geq \lambda_d$ be the largest d eigenvalues of B_L, $u_1, \dots, u_d$ the corresponding unit length eigenvectors, and $\lambda_i^+ = \max\{\lambda_i, 0\}$ for all $i = 1, \dots, d$. The vector $y_i \in \mathbb{R}^k$ with the coordinates of the k landmarks in the i-th dimension is

$$y_i = \sqrt{\lambda_i^+} u_i. \tag{3.45}$$

In a second step each of the $n - k$ non-landmark nodes is placed relatively to the k landmarks, using a result of Gower (1968). The vector $x_i \in \mathbb{R}^n$ with coordinates of all n nodes in the i-th dimension is given by

$$x_i = -\frac{1}{2\lambda_i^+} D_P^{(2)} J_k y_i = -\frac{1}{2\sqrt{\lambda_i^+}} D_P^{(2)} J_k u_i \ , \tag{3.46}$$

which, intuitively, places all non-landmark nodes in a weighted barycenter, with weights determined by their square distances to the k landmarks. Note that landmark nodes retain their original positions since each of them is already in the weighted barycenter of all $k - 1$ other landmarks. Landmark MDS is summarized in Algorithm 5. Figure 3.6 shows drawings of an example graph with Pivot and Landmark using the same pivot/landmark sets.

3.4 Related Methods

The analysis and visualization of proximity and similarity are common to many different areas of scientific application, either explicitly or implicitly. This has led to the development of data analysis techniques which have the same goal as MDS or use related mathematical tools. This section discusses how the speedup methods for classical scaling are related to these techniques.

In the multidimensional scaling literature this problem is often referred to as the analysis of *rectangular data*, as discussed by Heiser and Meulman (1983a).

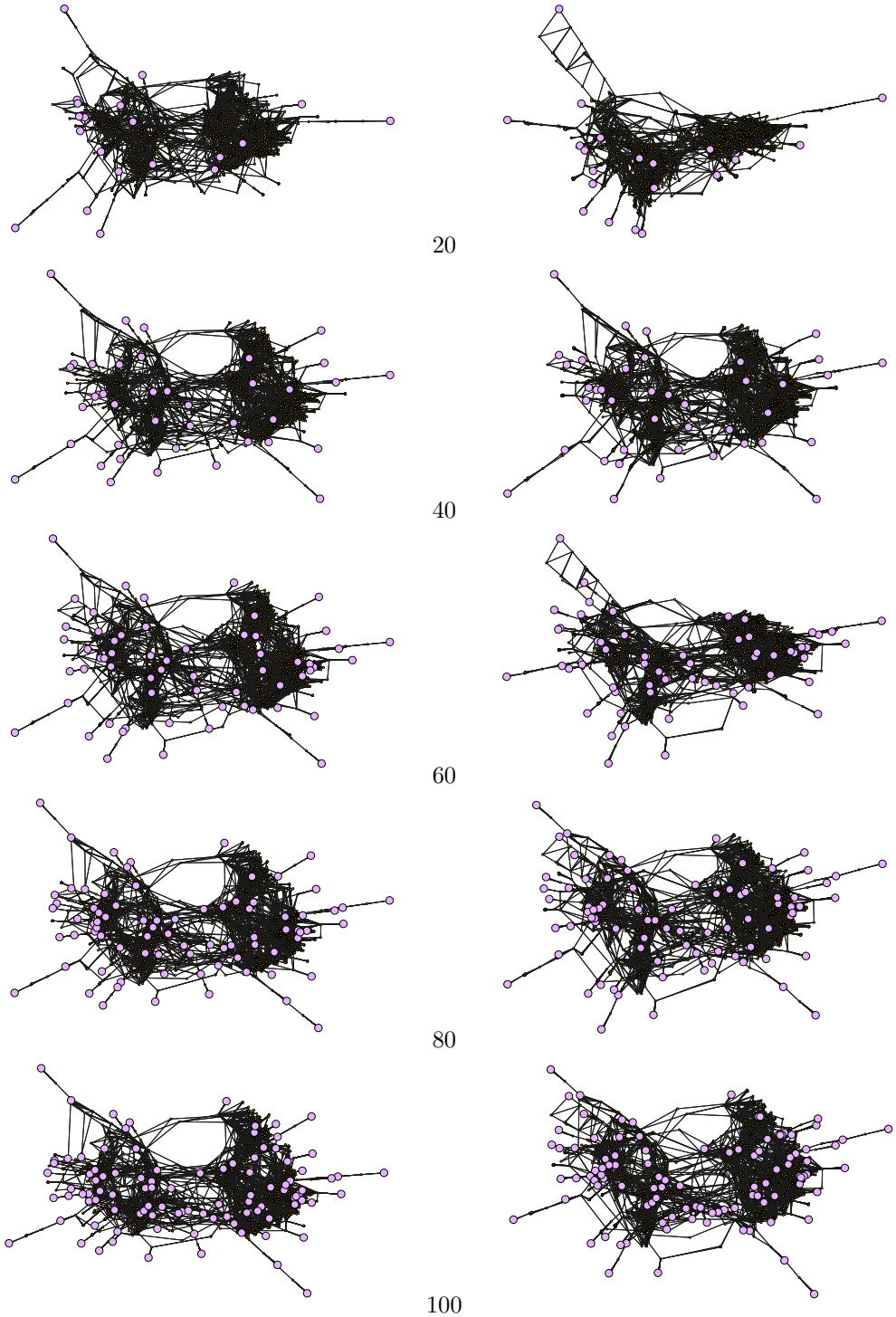


Figure 3.6: Pivot MDS vs. Landmark MDS for the esslingen1 graph, using the same incrementing sets of pivots/landmarks, which are highlighted by larger circles.

Algorithm 5: Landmark MDS

Input: undirected connected graph $G = (V, E)$, dimensionality $d \in \mathbb{N}$
Output: coordinate vectors $x_1, \ldots, x_d \in \mathbb{R}^n$

$D_P \leftarrow$ pivot matrix for G generated with Algorithm 3
$D_L \leftarrow$ landmark matrix (3.43) extracted from D_P
$(\lambda_1, u_1), \ldots, (\lambda_d, u_d) \leftarrow$ classical scaling on D_L using Algorithm 1
for $i = 1, \ldots, d$ **do**
  $x_i \leftarrow -\frac{1}{2\sqrt{\lambda_i}} \left(D_P^{(2)} J_k \right) u_i$

Table 3.1: Time and space complexities of classical scaling in a graph with n nodes and m edges and the speedup variants when using k landmarks/pivots, assuming a constant number of power iteration steps in the matrix decomposition.

method	memory	distance computation	decomposition
Classical MDS	$\Theta(n^2)$	$\mathcal{O}(n(m + n \log n))$	$\mathcal{O}(n^2)$
Landmark MDS	$\Theta(kn)$	$\mathcal{O}(k(m + n \log n))$	$\mathcal{O}(kn)$
Pivot MDS	$\Theta(kn)$	$\mathcal{O}(k(m + n \log n))$	$\mathcal{O}(k^2 n)$

The name characterizes non-square matrices, in which rows and columns correspond to disjoint sets of entities, which are to be positioned and analyzed either in separate column and row spaces or in a joint Euclidean space.

Where appropriate, rectangular data matrices may be interpreted as adjacency matrices of bipartite graphs, such as affiliation data. Nodes are of (at least) two types, and edges are only allowed between nodes of different types. For an application of classical scaling to scalable visualization of such bi- or multipartite graphs, see Chapter 9.

3.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is used in the analysis of multivariate data; see Joliffe (1986) for a comprehensive account. The input is a set of n data objects, each described by an m-tuple of *features*, hence a matrix $Y \in \mathbb{R}^{n \times m}$; without loss of generality, the entries in each column sum to zero. If the features in Y are real values, they can be directly interpreted as coordinates in the m-dimensional Euclidean space.

The objective of PCA is to reduce the dimensionality m of the original data to d *principal* axes ($d \leq m$) which explain as much of the structure of the data as possible. PCA transforms m correlated (linear dependent) variables into a smaller set of d uncorrelated (orthogonal) variables.

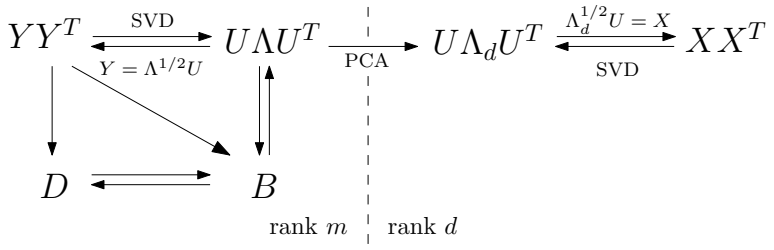


Figure 3.7: PCA on an m-dimensional coordinate matrix $Y \in \mathbb{R}^{n\times m}$ gives the same results as classical scaling on the Euclidean distances of the rows of Y in m-dimensional space.

The rotation to the *principal component space* of Y is obtained by the singular value decomposition

$$Y = U\Sigma V^T \tag{3.47}$$

where $U \in \mathbb{R}^{n\times n}$ contains the unit length eigenvectors of YY^T, $\Sigma \in \mathbb{R}^{n\times m}$ contains the singular values $\sigma_1, \ldots, \sigma_m$, and $V \in \mathbb{R}^{m\times m}$ the unit length eigenvectors of Y^TY. The rotation to the principal axes with the coordinate matrix $X \in \mathbb{R}^{n\times m}$ is given by

$$X = YV. \tag{3.48}$$

Since the matrix of inner products of the original coordinates in the m-dimensional space is YY^T, the d-dimensional classical scaling solution on the m-dimensional Euclidean distances and the d leftmost columns in X coincide. The magnitude of the singular value σ_i explains how much of the structure in the data is explained by the i-th principal axis. Classical scaling for reducing m-dimensional to d-dimensional coordinates was re-discovered as *principal coordinate analysis* by Gower (1966). This connection is illustrated in Figure 3.7.

In the context of PCA, the singular value decomposition in Pivot MDS corresponds to principal components of the pivot matrix D_P, and the rows of the matrix C in (3.37) can be interpreted as features describing the position of nodes with respect to the k selected pivots.

3.4.2 Unfolding

Multidimensional unfolding (Coombs, 1950) is closely related to MDS. The unfolding model deals with two distinct sets of objects U, V $(U \cap V = \emptyset)$ $|U| = n, |V| = m$ and the objective is to construct a joint representation for all $n+m$ objects. The input is a matrix of $n \cdot m$ dissimilarities or distances between every pair $(u, v), u \in U, v \in V$.

The term *unfolding* is motivated by the following intuition: Assuming that a solution is already available, a preference order with respect to a particular object is obtained by picking the configuration with one hand at that object's

position and folding it into the other hand like a handkerchief. The actually desired configuration is obtained by the reverse operation, i.e., by unfolding the folded handkerchief.

The connection to MDS is made by formulating the unfolding problem as a dissimilarity in a set of $n + m$ objects, with missing $n \times n$ and $m \times m$ blocks for the dissimilarities within U and V. A solution for the unfolding problem is obtained by MDS on the $(n + m) \times (n + m)$ matrix

$$\left[\begin{array}{c|c} ? & D^T \\ \hline D & ? \end{array}\right]$$

where "?" denotes a (square) block of missing within-set values.

In the MDS literature, unfolding is mostly done non-metrically, since the given dissimilarities are usually not considered to have Euclidean structure, but to be the result of subjective rankings. See Chapter 4 about treating missing values in MDS.

Schönemann (1970b) describes a metric version of unfolding, which uses the singular value decomposition of the double-centered matrix $-\frac{1}{2}J_n D^{(2)} J_k$ of squared dissimilarities, exactly as in Pivot MDS in (3.37). This metric version of unfolding is rarely used by itself, but mostly for initializing non-metric unfolding (Gold, 1973; Heiser and Meulman, 1983a; Nakanishi and Cooper, 2003).

The configurations resulting from metric unfolding are often considered problematic because there is no joint representation of both row and column objects in a joint space, since row and column space have no joint origin. However, this concern does not apply to Pivot MDS. Without loss of generality, let the k selected pivots $\{p_1, \ldots, p_k\}$ be represented by *column objects*, and all nodes be represented by *row objects*. Then every column object is also a row object. The row i and column j of a particular pivot p_j are connected by $d_{ij} = 0$ if $i = p_j$.

3.4.3 High Dimensional Embedding

Harel and Koren (2002b) propose a fast graph drawing algorithm called *high-dimensional embedding (HDE)*. In a first phase, the graph $G = (V, E)$ is embedded in a k-dimensional space. The coordinates are obtained by picking k pivots $p_1, \ldots, p_k$ and associating each of them with an axis in the space, and using the graph-theoretical distances $d(p_i, v)$ from this pivot p_i to all other nodes $v \in V$ as coordinates on this i-th axis. Writing the coordinates as a matrix $X \in \mathbb{R}^{k \times n}$ results exactly in the transpose of the pivot matrix D_P of Pivot MDS in (3.26). The strategy for picking pivots is the same as for Pivot MDS.

In a second phase, the axes in the high-dimensional drawing X are column-centered to $\hat{X}$ with

$$\hat{X} = X J_n, \tag{3.49}$$

where $J_n = I - \frac{1}{n}11^T$ is a column-centering matrix, so that the mean of each axis is zero. This centered k-dimensional configuration is subjected to PCA (see Section 3.4.1), computing the two dominant eigenvectors $u_1, u_2 \in \mathbb{R}^k$ of

$$S = \frac{1}{n}\hat{X}\hat{X}^T \in \mathbb{R}^{k\times k} \tag{3.50}$$

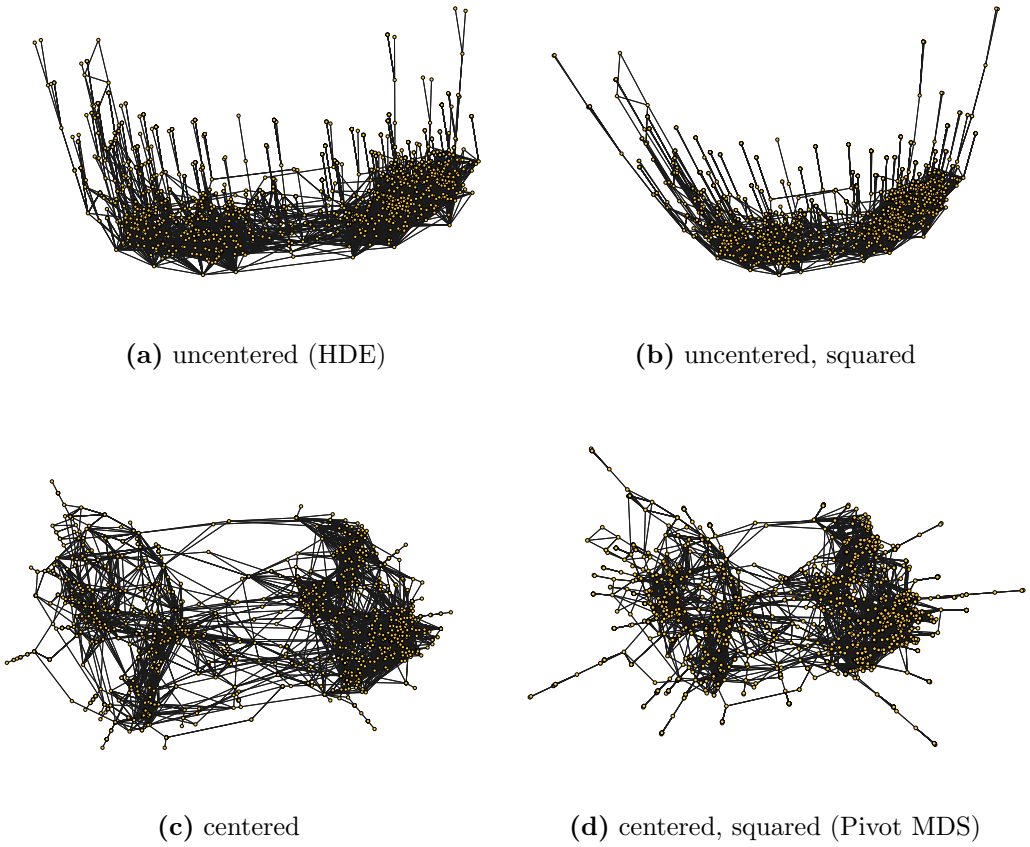
and using the projections

$$\hat{X}^T u_1, \hat{X}^T u_2 \in \mathbb{R}^n \tag{3.51}$$

as two-dimensional coordinates.

This approach is strikingly similar technically to Pivot MDS, in that the graph-theoretical distances from a pivot set are used. These distances are, however, neither squared nor double-centered, but directly interpreted as coordinates in a k-dimensional space. Both approaches are motivated by different intuitions and produce different results: HDE transforms k possibly correlated variables (the embedding X) into two uncorrelated variables (the layout) and interprets the matrix as coordinates, while Pivot MDS interprets the matrix directly as dissimilarities or distances to find coordinates.

Both HDE and Pivot MDS have the same running time complexity. Pivot MDS appears to yield drawings of superior quality for most input graphs, which is essentially due to two phenomena: First, the HDE layouts appear to be unevenly dense, with increasing density in peripheral regions of the configuration. This is not an issue in (Pivot) MDS because the dissimilarities are squared. Second, the skewed HDE layouts found for some types of graphs (Hachul and Jünger, 2006) can be attributed to the missing row-centering. Figure 3.8 illustrates these differences. For experiments and further applications see also Koren and Harel (2005).



(a) uncentered (HDE)

(b) uncentered, squared

(c) centered

(d) centered, squared (Pivot MDS)

Figure 3.8: HDE vs. Pivot MDS (50 pivots): The layouts of the esslingen1 graph resulting from (un)centered (squared) distances, illustrating the shortcomings of HDE.

Chapter 4

Distance Scaling

Although the classical scaling approach (see Chapter 3) produces optimal solutions, the underlying dissimilarity and Euclidean distance model is quite restricted. The advent of more powerful computing equipment stimulated the development of more flexible methods for dimensionality reduction. Some of the reasons for this development are situations where classical scaling is not appropriate:

- Classical scaling tries to fit distances to distances, but has to take a detour via inner products.
- The hypothesis that the given distances are Euclidean restricts the domain of possible inputs.
- It is not straightforward how classical scaling should deal with missing input values.
- Since the distances contribute quadratically to the strain measure, the fit of large distances is preferred over the fit of small distances.

In the early 1960s new MDS methods were developed that did not suffer from these drawbacks. They proved to be more flexible and intuitive than classical scaling. Most of the MDS approaches that have been created since that time try to arrange objects by directly fitting the distances; therefore, they will be termed *distance scaling* to distinguish them from the (classical) inner-product scaling.

The fitting is done by formulating the fit as an objective function involving the distances and devising an algorithm to minimize this function. Unlike classical scaling, no method is known for constructing algebraic optimal solutions. Instead, distance scaling resorts to iterative algorithms, which minimize some

objective function involving the distances to be fitted. Some of these iterative algorithms are similar or even identical to the ones used for graph layout as *force-directed* or *energy-based layout*. This chapter will focus on these similarities and identities and presents an extended MDS model that may be applied to frequently occurring graph layout problems.

4.1 Stress Minimization

4.1.1 Stress

The overall goal of multidimensional scaling is to fit the desired distances d_{ij} among n objects given in matrix form $D = (d_{ij}) \in \mathbb{R}^{n \times n}$ by the distances between d-dimensional coordinates $x_1, \ldots, x_n \in \mathbb{R}^d$, such that

$$d_{ij} = \|x_i - x_j\| \qquad \text{for all } i, j \in \{1, \ldots, n\}. \tag{4.1}$$

In general it is not possible to find coordinates $x_1, \ldots, x_n$ which exactly satisfy these requirements, unless the distances are derived from original Euclidean positions, which classical scaling then reconstructs, see Section 3.1.

When there are no such original positions, the amount of deviation from this goal is measured by an objective function, which incorporates the distances $\|x_i - x_j\|$, rather than inner products $\langle x_i, x_j \rangle$, as in classical scaling. The deviations are aggregated over all pairs of objects in the *(unweighted) stress*

$$\sigma(X) = \sum_{i<j} \left(d_{ij} - \|x_i - x_j\|\right)^2 \tag{4.2}$$

where the notation $\sum_{i<j}$ is shorthand for $\sum_{i=1}^{n} \sum_{j=1}^{i-1}$ (tacitly omitting n), the summation over all possible unordered pairs, and $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$ is a configuration matrix.

The smaller the stress value, the better the found configuration $x_1, \ldots, x_n$ fits the given distances. From an algorithmic point of view, distance scaling becomes an optimization problem in which (4.2) is to be minimized over all possible d-dimensional configurations:

$$\min_{X \in \mathbb{R}^{n \times d}} \sigma(X), \tag{4.3}$$

Thus, the term $(d_{ij} - \|x_i - x_j\|)^2$ is the individual contribution of the pair i, j to the entire error.

There are quite a few other fit measures for distance scaling, e.g. operating on squared distances (Takane et al., 1977). For a summary of fit measures and their relations see Heiser (1998).

The primary motivation for introducing distance scaling was *non-metric scaling* (Shepard, 1962), where the goal is to fit the *rank-order* of distances, rather than their *values*, as in *metric scaling*. The rank-order is formulated as monotonicity constraints

$$d_{ij} \leq d_{kl} \Rightarrow \|x_i - x_j\| \leq \|x_k - x_l\| \qquad \text{for all } i, j, k, l \in \{1, \ldots, n\}.$$

Non-metric scaling is done by replacing the dissimilarities (d_{ij}), which are not required to form a metric, by *disparities* ($\hat{d}_{ij}$) in (4.2). The disparities serve as metric auxiliary dissimilarities, which are not fixed but dynamically updated in an optimization process.

The origin of non-metric scaling lies in the field of psychometrics, where the analysis is more often interested in the rank-order of subjects and stimuli than in the actual values because the scale of the input dissimilarities is meaningless.

In fact, distance scaling is mostly done non-metrically in psychometrics. This has led to the occasional misleading distinction of classical scaling and distance scaling as "metric and non-metric", even though both of them are actually metric, and distance scaling becomes non-metric only when monotonicity constraints are introduced. Instead, distance scaling without monotonicity constraints is more correctly characterized as *absolute scaling* or *metric distance scaling*.

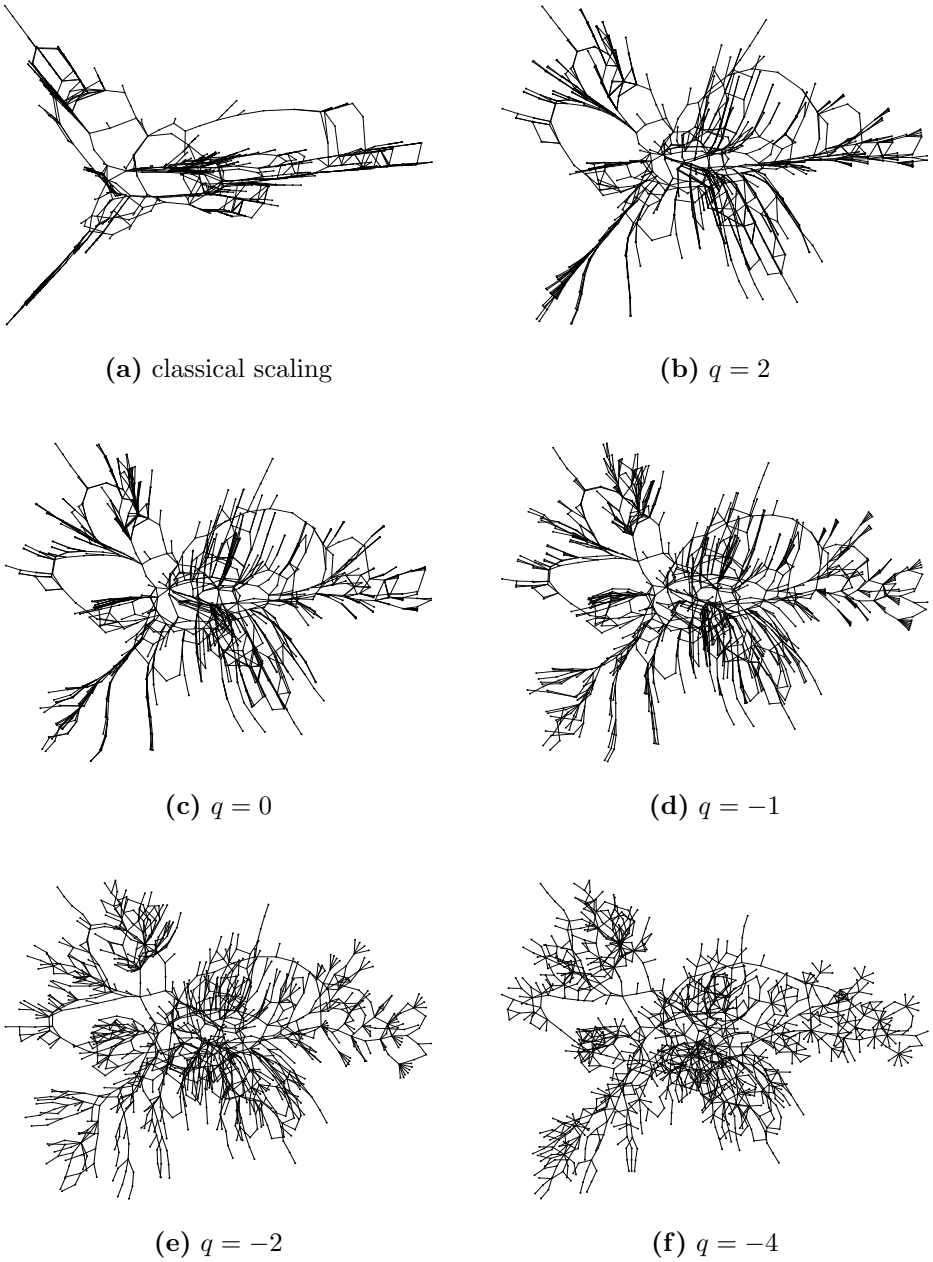
4.1.2 Weights

The stress measure is quite sensitive to outliers in the input due to measurement errors (Graef and Spence, 1979). The influences of particular object pairs to the stress are adjusted by introducing weights $w_{ij} \geq 0$ into the stress as coefficients of the individual error terms

$$\sigma(X) = \sum_{i<j} w_{ij} \left(d_{ij} - \|x_i - x_j\|\right)^2. \tag{4.4}$$

Most commonly, weights are a function of the dissimilarities, e.g., an exponential weighting scheme $w_{ij} = d_{ij}^q$ for some $q \in \mathbb{R}$. Weights make the MDS framework more flexible and more general. Some other independently developed data analysis methods were found to be members of the MDS family:

- $w_{ij} = 1$: In the original proposals of MDS, no weights were used, and all pairs are implicitly assigned the same weights (Kruskal, 1964; Shepard, 1962).
- $w_{ij} = d_{ij}$: Weights are proportional to their magnitude, giving large dissimilarities more impact on the stress than small ones.



(a) classical scaling

(b) $q = 2$

(c) $q = 0$

(d) $q = -1$

(e) $q = -2$

(f) $q = -4$

Figure 4.1: Example drawings for the 1138bus graph (1138 nodes, 1458 edges). (a) is generated with classical scaling and used as the initial solution for (b)–(f) computed with distance scaling and weights $w_{ij} = d_{ij}^q$.

- $w_{ij} = d_{ij}^{-1}$: Sammon (1969) proposed a "nonlinear technique" for pattern analysis, which turned out to be an instance of multidimensional scaling (Kruskal, 1971). The quality of the fit small target distances is increased compared to the fit of larger dissimilarities, by using weights inversely proportional to the dissimilarities.

- $w_{ij} = d_{ij}^{-2}$: This weighting scheme was introduced in *elastic scaling* by McGee (1966). Instead of fitting absolute values by minimizing *absolute* residual error terms
$$(d_{ij} - \|x_i - x_j\|)^2$$
the goal is to achieve a fit of the distance ratios, expressed by *relative* error terms
$$\left(1 - \frac{\|x_i - x_j\|}{d_{ij}}\right)^2 = \frac{1}{d_{ij}^2}(d_{ij} - \|x_i - x_j\|)^2$$
resulting in weights d_{ij}^{-2}; McGee termed the correspondingly weighted stress *work*.

- $w_{ij} = e^{-d_{ij}}$: The influence of very large dissimilarities is down-weighted similarly to the above weightings.

An important application of weights is the treatment of *missing values*: When the desired distance is not known or unreliable for a pair, this pair is simply ignored by setting $w_{ij} = 0$. DeSarbo and Rao (1984) use weights to avoid trivial solutions. A general discussion of weighting schemes is given by Heiser (1998). In the context of graphs, other weighting schemes and dissimilarities are discussed in (Buja and Swayne, 2002; Cohen, 1997).

Figure 4.1 illustrates the use of different weighting schemes for drawing a graph. See Chapter 8 for an application in which several weight matrices are used to convey different analytic aspects of a graph in radial layouts.

4.1.3 Stress Majorization

Since no closed form for a global minimum of (4.4) is known, all of the approaches for finding a configuration X with low stress value $\sigma(X)$ work iteratively. In the early days of distance scaling the stress function was minimized by *gradient descent* (Guttman, 1968; Kruskal, 1964). Using partial derivatives of the stress function, a direction is determined for moving objects to a new position so as to lower the stress. Empirically, this strategy has proven to be satisfactory for most purposes, but is difficult to implement correctly and does not always converge.

This approach was superseded by *majorization* (de Leeuw, 1977). Essentially, instead of minimizing the stress itself directly, another function is minimized

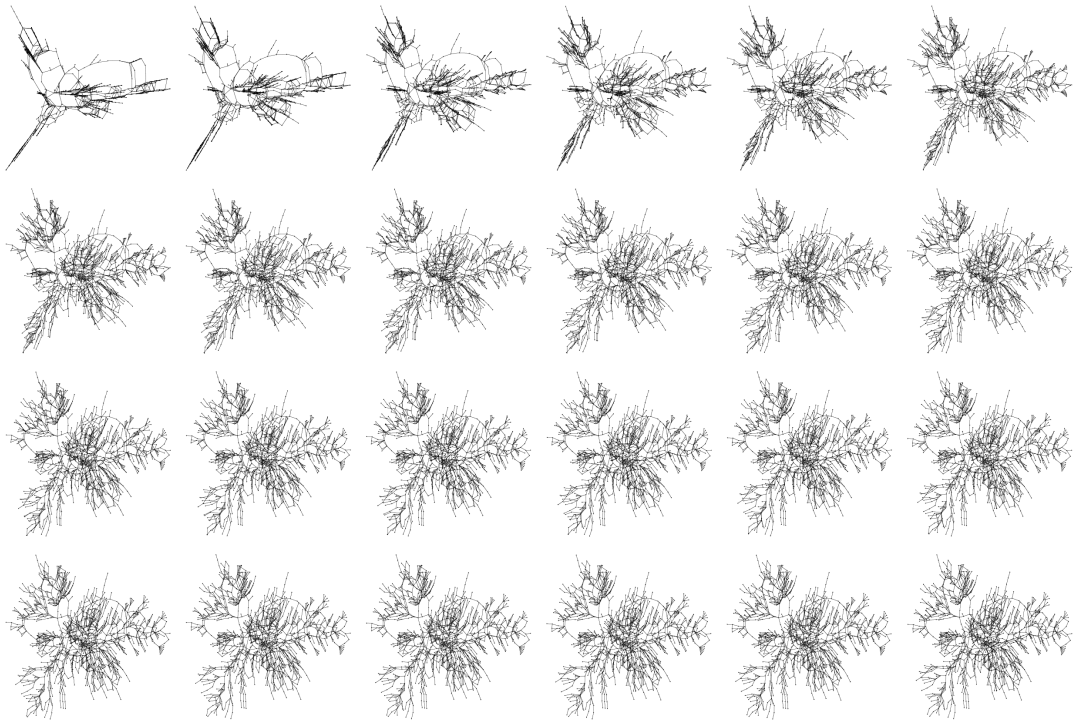


Figure 4.2: Sequence of layouts for the 1138bus graph after each majorization, using weights $w_{ij} = d_{ij}^{-2}$. The iteration is initialized by classical scaling.

which majorizes the stress, i.e., is greater or equal, but easier to minimize algebraically and computationally. Majorization is currently the method of choice for solving stress-related problems, particularly because it is more robust and has fewer parameters than gradient descent, e.g. step lengths, and provably converges.

In the following, it is assumed without loss of generality that $\sum_{i,j} w_{ij} d_{ij}^2 = 1$, which is achieved by multiplying all weights w_{ij} as necessary. The stress term (4.4) is expanded to

$$\begin{aligned} \sigma(X) &= \sum_{i,j} w_{ij} d_{ij}^2 + \sum_{i,j} w_{ij} \|x_i - x_j\|^2 - 2 \sum_{i,j} w_{ij} d_{ij} \|x_i - x_j\| \\ &= 1 + \sum_{i,j} w_{ij} \|x_i - x_j\|^2 - 2 \sum_{i,j} \frac{w_{ij} d_{ij}}{\|x_i - x_j\|} \langle x_i - x_j, x_i - x_j \rangle. \end{aligned}$$

It can be shown that for any configuration $Y = [y_1, \ldots, y_n]^T \in \mathbb{R}^{n \times d}$ the

majorization inequality

$$\begin{aligned}\sigma(X) &= 1+\sum_{i,j} w_{ij}\|x_i-x_j\|^2 - 2\sum_{i,j}\frac{w_{ij}d_{ij}}{\|x_i-x_j\|}\langle x_i-x_j, x_i-x_j\rangle \\ &\leq 1+\sum_{i,j} w_{ij}\|x_i-x_j\|^2 - 2\sum_{i,j}\frac{w_{ij}d_{ij}}{\|y_i-y_j\|}\langle x_i-x_j, y_i-y_j\rangle \quad =: \quad \tau(X,Y)\end{aligned}$$

holds, and that $\tau(X,Y)$ is a majorizing function which bounds $\sigma(X)$ from above, with equality if $X=Y$. The advantage of minimizing $\tau(X,Y)$ over X with fixed Y is that, unlike $\sigma(X)$, it is a convex function and thus has a global minimum.

In the iterative majorization procedure the fixed Y is the current configuration, denoted by $X^{[t]}$. The goal is to find a configuration X which globally minimizes $\tau(X,X^{[t]})$, in which case

$$\sigma(X) \leq \tau(X,X^{[t]}) \leq \tau(X^{[t]},X^{[t]}) = \sigma(X^{[t]})$$

holds, and the stress is non-increasing. In the next step, the configuration is updated by $X^{[t+1]} := X$, after which the minimization step is repeated, and so on.

In a step of the iterative majorization process, the desired minimizer X can be computed by matrix inversion (de Leeuw, 1977; Guttman, 1968). An alternative way of computing the successive configuration with non-increasing stress is due to Gansner et al. (2004). For every object i new coordinates $x_i^{[t+1]}$ are computed from the current coordinates $x_i^{[t]}$, while the other $n-1$ objects are fixed. The update is

$$x_i^{[t+1]} \leftarrow \frac{\sum_{j\neq i} w_{ij}\left(x_j^{[t]} + s_{ij}^{[t]}\cdot(x_i^{[t]}-x_j^{[t]})\right)}{\sum_{j\neq i} w_{ij}} \tag{4.5}$$

where

$$s_{ij}^{[t]} = \begin{cases} \frac{d_{ij}}{\|x_i^{[t]}-x_j^{[t]}\|} & \text{if } \|x_i^{[t]}-x_j^{[t]}\|>0 \\ 0 & \text{otherwise;}\end{cases} \tag{4.6}$$

this is repeated until the configuration is stable when the relative change of the configuration is below some threshold value $\epsilon>0$, i.e.,

$$\frac{\|X^{[t]}-X^{[t+1]}\|}{\|X^{[t]}\|} < \epsilon$$

or the stress cannot be reduced further significantly,

$$\frac{\sigma(X^{[t]})-\sigma(X^{[t+1]})}{\sigma(X^{[t]})} < \epsilon.$$

Algorithm 6: Stress majorization

Input: distances $D = (d_{ij})$, weights $W = (w_{ij})$, initial configuration $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$, accuracy $\epsilon > 0$

Output: final positions x_i for all $i \in \{1, \ldots, n\}$

while *relative change of* X *is more than* ϵ **do**
  foreach $i \in \{1, \ldots, n\}$ **do**
    $\hat{x}_i \leftarrow \dfrac{\sum_{j \neq i} w_{ij}\big(x_j + s_{ij}(x_i - x_j)\big)}{\sum_{j \neq i} w_{ij}}$
  foreach $i \in \{1, \ldots, n\}$ **do**
    $x_i \leftarrow \hat{x}_i$

Alternatively, the computation may simply be terminated after a predefined number of steps or period of computation time.

The generated sequence $X^{[0]}, X^{[1]}, X^{[2]}, \ldots, X^{[t]}$ of layouts has non-increasing stress, $\sigma(X^{[0]}) \geq \sigma(X^{[1]}) \geq \sigma(X^{[2]}) \geq \cdots \geq \sigma(X^{[t]})$, and converges to a local minimum (de Leeuw, 1988). Figure 4.2 shows such a sequence for an example graph. Pseudocode is given in Algorithm 6.

The iteration has to be initialized with a starting solution $X^{[0]}$. A frequently used initialization strategy is classical scaling. Experimental evidence for this, specifically when applied to graph drawing, is given in Chapter 5; for more general contexts, see also Malone et al. (2002).

4.2 Related Methods

Most applications of MDS to graph drawing set the desired distances to be the shortest-path distances in the graph, which often spread nodes well over the drawing and display symmetries and clusterings.

Kruskal and Seery (1980) are among the first to explicitly apply MDS to shortest-paths distances. While they state that "for the application in this paper, the so-called 'classical' method of MDS is entirely adequate", their graph drawings are generated with a stress minimization program available to them.

In the following years some methods have been devised that implicitly make use of ideas and tools quite related to distance scaling. Weighting schemes and alternatives to shortest-path distances are given by Cohen (1997). Using the stress majorization approach by de Leeuw (1977) was made popular in the graph drawing community by Gansner et al. (2004).

The principal objective common to all *force-directed* or *organic* layouts is that adjacent nodes should be located close to each other, or even at the same

place. For an undirected graph $G = (V, E)$, this can be cast, similarly to stress, as finding positions $p(v)$ for all nodes $v \in V$ minimizing

$$\sum_{\{u,v\}\in E} \|p(u) - p(v)\|^2. \tag{4.7}$$

Without modification, solving this problem is trivial, since a layout in which all nodes collapse to the same point reduces this sum to zero. Different strategies can be followed to avoid these undesired solutions (Brandes, 2001). In the following, these will be recast in the framework of stress minimization. For an elaborate discussion of the connections between MDS and graph layout, see also de Leeuw and Michailidis (2000).

4.2.1 Barycenter Layout

To prevent the nodes from collapsing, the positions of a subset $V' \subseteq V$ of nodes are fixed, and all others are allowed to move (Tutte, 1963) freely. The term to be minimized is

$$\sum_{v\in V\setminus V'} \sum_{u\in N(v)} \|p(u) - p(v)\|^2 \tag{4.8}$$

subject to the fixed positions of nodes in V', and the ideal distance between a pair of adjacent nodes is zero. A layout with this objective is done by iteratively placing every node in the barycenter of its neighbors until the configuration is stable.

4.2.2 Spectral Layout

One of the problems with this method is that for general undirected graphs it is not clear which nodes should be fixed. An alternative approach is due to Hall (1970): When $X = [p(v_1), \ldots, p(v_n)]^T \in \mathbb{R}^{n\times d}$ is the configuration matrix of node positions, (4.7) can be rewritten as

$$\sum_{\{u,v\}\in E} \langle p(u) - p(v), p(u) - p(v)\rangle = \mathrm{tr}(X^T L X), \tag{4.9}$$

where $L = \mathrm{diag}(\deg(v_1), \ldots, \deg(v_n)) - A$ is the *Laplacian matrix* and A is the adjacency matrix of G. The trivial undesired solution with collapsing nodes corresponds to the eigenvector $\frac{1}{\sqrt{n}}[1, \ldots, 1]^T$ of L associated with the eigenvalue zero; this trivial solution is avoided by constraining the solution to be orthogonal to this eigenvector. Thus, the d-dimensional layout obtained by using the d smallest non-zero eigenvalues of L minimizes (4.7). For more details and alternative derivations of the Laplacian layout, see Koren (2003). An interesting

connection between this method and MDS, established by Fouss et al. (2005) and Fleischer (2007), is that Laplacian layout is equivalent to classical scaling using as the input distance the square root of the effective resistance between two nodes in the corresponding electrical network.

4.2.3 Force-directed Layout

The desire of placing as close as possible to their neighbors is equivalent to a desired edge length of zero. Using some non-zero edge length instead is the main objective of *force-directed* algorithms. They model the graph as a system of physical objects and forces, and the movement of objects is simulated numerically. Edges are modeled by springs which strive at having a certain ideal length.

On every node $v \in V$ a force is exerted, described by a d-dimensional net force vector $f(v) \in \mathbb{R}^d$ with

$$f(v) = \sum_{u \in N(v)} f_a(u,v) - \sum_{u \in V \setminus \{v\}} f_r(u,v), \tag{4.10}$$

where the $f_a(u,v) \in \mathbb{R}^d$ are attractive forces of springs between adjacent nodes, which try to attain an ideal length and $f_r(u,v) \in \mathbb{R}^d$ are repulsive forces between all pairs of nodes. Iteratively, all nodes are moved into the direction of their force by a small step until the configuration does not change significantly.

In the original spring algorithms the choice of forces was ad-hoc (Eades, 1984; Fruchterman and Reingold, 1991). In the algorithm described by Di Battista et al. (1998) it is assumed that the springs follow Hooke's law and that electrical forces are used for repulsion. The forces used in (4.10) are

$$\begin{aligned} f_a(v,u) &= w_{uv} \cdot (d_{uv} - \|p(u) - p(v)\|) \cdot \frac{p(v) - p(u)}{\|p(v) - p(u)\|} \\ f_r(v,u) &= \frac{z_{uv}}{\|p(v) - p(u)\|^2} \cdot \frac{p(v) - p(u)}{\|p(v) - p(u)\|} \end{aligned}$$

where $d_{uv} > 0$ is the desired length of edge $\{u,v\}$, $w_{uv} \geq 0$ is a stiffness parameter of the corresponding spring, and $z_{uv} \geq 0$ is a parameter for the strength of the electrical field. The energy of the entire physical system is

$$\sum_{u,v} w_{uv}(d_{uv} - \|p(u) - p(v)\|)^2 + \sum_{u,v} \frac{z_{uv}}{\|p(u) - p(v)\|}. \tag{4.11}$$

In the framework of MDS, this corresponds to stress restricted to adjacent node pairs, where the desired distances and weights are set to

$$d_{uv} = w_{uv} = \begin{cases} 1 & \text{if } \{u, v\} \in E \\ 0 & \text{if } \{u, v\} \notin E \end{cases}$$

with an additional repulsion penalty term which serves to scatter the nodes over the drawing. See de Leeuw and Michailidis (2000) for an algorithm in which the iterative simulation of spring forces is replaced by a converging majorization algorithm. Alternative force models tuned for the discovery and the display of clusterings are discussed by Noack (2007).

Force directed methods are notoriously slow, tend to get stuck in local minima, and a good initial solution is crucial. Various strategies have been presented to tackle these problems (Gajer et al., 2000; Hadany and Harel, 2001; Harel and Koren, 2002a; Walshaw, 2000). Particularly, the algorithm by Gajer and Kobourov (2001) uses a multi-level hierarchy of constructing, combining, and postprocessing initial layouts. Hachul and Jünger (2004) present a spring embedder with a multi-level force calculation scheme; the experimental study in Chapter 5 compares results of MDS and these two algorithms.

4.2.4 Energy-based Layout

Kamada and Kawai (1989) introduce springs for non-adjacent nodes with desired lengths proportional to the shortest path connecting them, instead of repulsive forces to keep non-adjacent nodes apart. The energy to be minimized is equal to (4.4) setting $w_{ij} = d_{ij}^{-2}$; the gradient terms are the same as the ones used by McGee (1966).

Following an idea from Quinn and Breuer (1979), updating the positions is carried out by allowing only one single node at a time to move, while keeping all others fixed. Gansner et al. (2004) show that the convergence properties of majorization still hold. Pseudo-code is given in Algorithm 7. The quantity s_{uv} for two nodes $u, v \in V$ is defined as

$$s_{uv} = \begin{cases} \frac{d_{uv}}{\|p(u) - p(v)\|} & \text{if } \|p(u) - p(v)\| > 0 \\ 0 & \text{otherwise} \end{cases}$$

similar to (4.6).

4.3 Weak Constraints

When structural information is available about the objects described by the dissimilarities, it can be integrated in the distance scaling process by imposing

Algorithm 7: Graph drawing by stress majorization

Input: undirected connected graph $G = (V, E)$, initial positions $p(v)$ for all $v \in V$, accuracy $\epsilon > 0$

Output: positions $p(v)$ for all $v \in V$

$D \leftarrow (d_{uv})$ // shortest-path distances

$W \leftarrow (w_{uv}) = (d_{uv}^{-2})$

while *p changes by more than ϵ* **do**

  foreach $v \in V$ **do**

    $$p(v) \leftarrow \frac{\sum_{u \in V \setminus \{v\}} w_{uv}\big(p(u) + s_{uv}(p(v) - p(u))\big)}{\sum_{u \in V \setminus \{v\}} w_{uv}}$$

constraints on the solutions. For example, consider a social network in which every actor has a group label attached. Without constraints, a layout is simply given by distance scaling only on the shortest-path distances among the nodes. In addition, the desire to place nodes in the same group close together and to keep nodes in different groups separated can be expressed as additional proximities, which are added to the layout as *distance constraints*.

In the MDS literature two different constraint approaches are distinguished (Borg and Groenen, 2005):

- *Strong constraints:* Only solutions are feasible in which all constraints are satisfied. Among these, a configuration with minimal stress is sought.
- *Weak constraints:* Solutions are allowed to deviate from the set of constraints. An additional term in the objective function penalizes the amount of this deviation.

A configuration for which the additional penalty term has value zero also satisfies the constraints in the strong sense. When such a configuration exists, the constraints are called *realizable*. Note that, alternatively, these two classes of contraints are also frequently termed *hard* and *soft*.

For both of these approaches modifications of the distance scaling methods are available. This section concentrates on weak constraints, since a variety of graph layout problems can be expressed with them, and only minor modifications to existing algorithms are necessary.

MDS with weak constraints goes back to Borg and Lingoes (1980). A comprehensive survey of constrained and confirmatory MDS is given by Heiser and Meulman (1983b). Weak constraints are considered by ten Berge (1991) in the context of related regression problems.

Distance scaling methods for strong constraints are discussed by de Leeuw and Heiser (1980). Strongly constrained MDS is applied to graph drawing for a given partition into vertical levels (Dwyer and Koren, 2005) and separation constraints (Dwyer et al., 2006b).

4.3.1 A Modified Stress Model

The stress in (4.4) is extended by another stress term involving the weak constraints. The optimization problem then becomes minimizing the convex combination

$$(1-\alpha)\cdot\sigma_{D,W}(X)+\alpha\cdot\sigma_{D',W'}(X). \tag{4.12}$$

The second component is called *penalty* for how far X is away from satisfying the constraints; $D=(d_{ij}), W=(w_{ij})\in\mathbb{R}^{n\times n}$ are the distance and weight matrices defined as in the unconstrained stress, $D'=(d'_{ij}), W'=(w'_{ij})\in\mathbb{R}^{n\times n}$ are the distance and weight matrices expressing the weak constraints, and the interpolation parameter $\alpha\in[0,1]$ determines the impact of the constraint penalty term $\sigma_{D',W'}(X)$; the constraints are ignored if $\alpha=0$, while the distances do not contribute to (4.12) if $\alpha=1$.

Chapter 5 describes an application of weakly constrained distance scaling to radial graph layout, in which $D=D'$ and the interpolation only involves the weight matrices.

4.3.2 Computation

To iteratively minimize the linear combination (4.12), Borg and Lingoes (1980) use the linear combination of the two individual update terms. Carried over to stress majorization, the update term for an individual position x_i of object i is

$$\hat{x}_i\leftarrow(1-\alpha)\cdot\frac{\sum_{j\neq i}w_{ij}\big(x_j+s_{ij}(x_i-x_j)\big)}{\sum_{j\neq i}w_{ij}}+\alpha\cdot\frac{\sum_{j\neq i}w'_{ij}\big(x_j+s'_{ij}(x_i-x_j)\big)}{\sum_{j\neq i}w'_{ij}} \tag{4.13}$$

where, as in (4.6),

$$s_{ij}=\begin{cases}\frac{d_{ij}}{\|x_i-x_j\|} & \text{if } \|x_i-x_j\|>0\\ 0 & \text{otherwise.}\end{cases}\quad\text{and}\quad s'_{ij}=\begin{cases}\frac{d'_{ij}}{\|x_i-x_j\|} & \text{if } \|x_i-x_j\|>0\\ 0 & \text{otherwise.}\end{cases}$$

In principle, a solution to a strongly constrained problem as defined in (4.12) may be found by setting $\alpha=1$ and thus directly minimizing $\sigma_{D',W'}(X)$, e.g. using Algorithm 6. However, this may lead to trivial solutions, which satisfy the constraints, but do not consider the distances (d_{ij}) at all.

Algorithm 8: Stress majorization with weak constraints

Input: distances $D = (d_{ij}), D' = (d'_{ij})$, weights $W = (w_{ij}), W' = (w'_{ij})$, initial positions $x_1, \ldots, x_n \in \mathbb{R}^d$, number $k \in \mathbb{N}$ of iterations
Output: final positions $x_1, \ldots, x_n \in \mathbb{R}^d$

for $\alpha = 0, \frac{1}{k}, \frac{2}{k}, \ldots, 1$ **do**
  foreach $i \in \{1, \ldots, n\}$ **do**
    $\hat{x}_i \leftarrow (1-\alpha) \cdot \frac{\sum_{j \neq i} w_{ij}\big(x_j + s_{ij}(x_i - x_j)\big)}{\sum_{j \neq i} w_{ij}} + \alpha \cdot \frac{\sum_{j \neq i} w'_{ij}\big(x_j + s'_{ij}(x_i - x_j)\big)}{\sum_{j \neq i} w'_{ij}}$
  foreach $i \in \{1, \ldots, n\}$ **do**
    $x_i \leftarrow \hat{x}_i$

To avoid this problem, the constraints are carefully introduced by gradually increasing the influence of the penalty. In a first phase a configuration is computed which minimizes just $\sigma_{D,W}(X)$. This configuration then serves as initial solution to a second phase, in which α is increased, thus granting the constraints more and more control over the optimization problem. While eventually arriving at $a = 1$ gives the constraints total control, it may also make sense to use a more moderate form of weak constraints by trading them off with distances, e.g., by finally setting $\alpha = \frac{1}{2}$.

Technically, in each iteration step a different optimization problem is sought to be solved; the initial solution for each of these problems is determined by what is obtained in the preceding step. Thereby, the iteration makes use of the sensitivity to initial solutions, which is commonly observed in iterative MDS algorithms; this is elaborated in the experimental study in Chapter 5.

It should be noted that, even though the component of $\sigma_{D,W}(X)$ vanishes when $\alpha \to 1$, minimizing (4.12) is *not* the same as directly minimizing $\sigma_{D',W'}(X)$.

This continuous shift from an unconstrained to a constrained problem tends to preserve the characteristic features of the initial solution resulting from the first phase. As a byproduct, the sequence of layouts which results from increasing α from 0 to 1 tends to appear as continuous and may be used for display, e.g., when animating dynamic graphs.

4.3.3 Applications

Weakly constrained distance scaling can be used for many applications. The concrete definition of D', W' is specific to the context. The following is just an enumeration of examples for constraints that may be expressed as distances; two of them are explicated in the end of this section, and in Chapter 8, respectively.

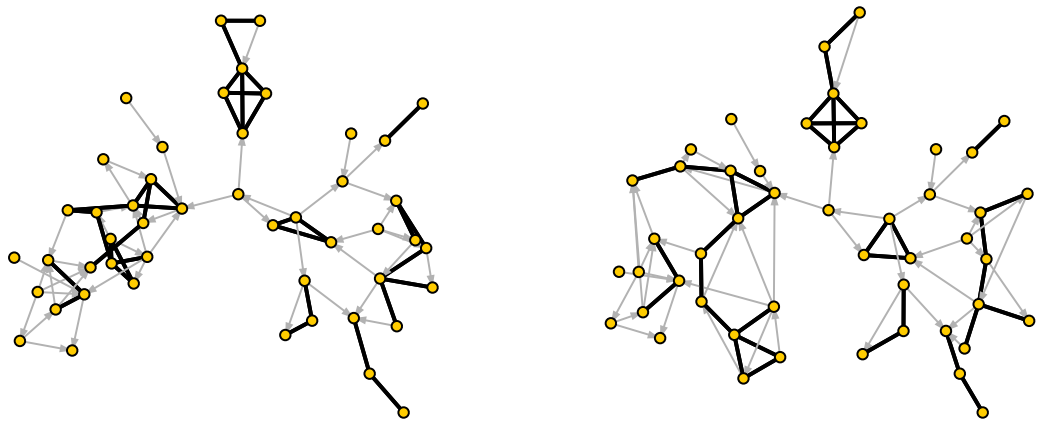
- *Cluster layouts:* In addition to desired distances between nodes, desired distances between and within groups are defined.
- *Radial drawings:* Nodes are constrained to lie on concentric circles with given radii, using an additional dummy node representing the center of these circles. See Chapter 8.
- *Layered drawings:* Nodes are constrained on a set of parallel lines, e.g., vertical coordinates.
- *Overlap removal:* When nodes are rendered as rectangles, the overlap is removed by introducing additional edges whose desired length is increased until the rectangles of the incident nodes no longer intersect (Gansner and Hu, 2009).
- *Evolving graphs:* A sequence of graphs $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ represents the evolution of a graph. Continuous dynamic layouts are obtained by interpolation using the accordingly defined distance matrices $D_1, \ldots, D_k$ and weight matrices $W_1, \ldots, W_k$. Special care has to be taken when graphs become disconnected. See Ambrosi and Hansohm (1987) for a more detailed discussion of *Dynamic MDS*.
- *Disconnected graphs:* Connected components are usually laid out individually, and put together compactly. Alternatively, when the connected components are related in some way, they can be constrained to be placed close to each other. An example will discussed in some more detail in the following.

Networks occurring in the social sciences frequently have reciprocated and non-reciprocated edges. For example, when ties in a social network are obtained by filling questionnaires, actor u may name v as a friend, but j does not necessarily reciprocate this friendship relation.

It is possible to compute a layout for this type of mixed graphs governed by confirmed edges. Two undirected graphs are derived from a directed weakly connected graph $G = (V, E)$, for which the distance and weight matrices are computed:

$$\begin{aligned} G_{\text{unconfirmed}} &= \big(V, \{\{u, v\} \subseteq V \colon (u, v) \in E \text{ or } (v, u) \in E\}\big) \\ G_{\text{confirmed}} &= \big(V, \{\{u, v\} \subseteq V \colon (u, v) \in E \text{ and } (v, u) \in E\}\big) \end{aligned}$$

$G_{\text{unconfirmed}}$ is just the underlying undirected graph of G and $G_{\text{confirmed}}$ is the underlying undirected graph G after all non-reciprocated edges have been deleted. Note that, while $G_{\text{unconfirmed}}$ is connected, $G_{\text{confirmed}}$ may become disconnected.



(a) unconstrained, $\alpha = 0$

(b) constrained by distances in $G_{\text{confirmed}}$, $\alpha = 1$

Figure 4.3: A social network of 48 nodes with 35 confirmed edges (thick, black, no arrow) and 46 unconfirmed edges (thin, gray, arrows). The unconstrained layout in (a) becomes the constrained layout in (b) by gradually increasing α to 1, thus reducing the impact of unconfirmed edges to zero.

This situation is not problematic in weakly constrained MDS when the involved matrices are set accordingly. The interpolation starts with

$$\begin{aligned} D &= (d_{uv}) = D(G_{\text{unconfirmed}}) \\ W &= (w_{uv}), w_{uv} = d_{uv}^{-2} \end{aligned}$$

and ends with the constraint matrices

$$\begin{aligned} D' &= (d'_{uv}) = D(G_{\text{confirmed}}) \\ W' &= (w'_{uv}) \text{ where } w'_{uv} = \begin{cases} d_{uv}^{-2} & \text{if } 0 < d'_{uv} < \infty, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Note that D' may contain infinite distances between nodes in different connected components. However, these entries are ignored because the corresponding entries in the weight matrix W' are zero. Figure 4.3 shows a social network with confirmed and unconfirmed edges.

Chapter 5

Experimental Study

Graph drawing is concerned with the geometric representation of graphs. For general undirected graphs, force-directed and energy-based layout algorithms are commonly used, because they are often easy to implement and experience shows that they can result in undistorted and readable layouts which reveal structural features such as local clustering and symmetry (Brandes, 2001).

Based on experimental evidence we argue that approximate classical scaling with subsequent stress reduction should be used instead. The requirements leading to this argument are:

1. *quality*: pairwise distances between vertices are represented well,
2. *scalability*: the algorithm scales to very large graphs, and
3. *simplicity*: the algorithm is easy to understand and implement.

Note that the quality criterion is implicit in force-directed algorithms. Classical scaling (Chapter 3) and stress minimization (Chapter 4) are instances of the general concept of multidimensional scaling. MDS of graph-theoretic distances has been used early on for automatic layout of social networks (Kruskal and Seery, 1980), without explicit reference in the well-known algorithm of Kamada and Kawai (1989), and in the wider context of data analysis (Buja and Swayne, 2002; Freeman, 2000), but the use of advanced MDS algorithms well-known in other fields has gained momentum only after Gansner et al. (2004) applied majorization to stress minimization in graph drawing. Stress minimization is generally assumed to be the method of choice for drawing general graphs, because of its intuitive and adaptable objective function and the visually pleasing layouts obtained. Yet, it is often found to be difficult to implement efficiently, and the presence of local minima is a serious concern.

Our study provides an assessment of layout quality and efficiency, and also yields a recommendation on how to implement the method to achieve reliability,

efficiency, and simplicity at the same time. While a considerable number of experimental studies have been conducted to assess graph drawing criteria and algorithm performance, only two are closely related (Brandenburg et al., 1996; Hachul and Jünger, 2006). However, these compare implementations of suites of related algorithms which are treated as black boxes. The combination of our in-depth study with these more general comparisons provides additional support for our conclusion.

A methodological contribution of our study is the design of experiments along explicit hypotheses about the performance of algorithms. These guided our choice of experiments and structure argumentation.

5.1 Hypotheses

A combination of theoretical properties, previous experience, popular beliefs, and preliminary tests, led us to formulate and test the hypotheses below. These shall not be read as if they were results, but serve to focus attention and are formulated in such a way that they can be tested with algorithmic experiments. We therefore conducted a series of experiments described in the next section. See Section 5.4 for a discussion of the results.

The first hypothesis basically rules out force-directed methods.

Hypothesis 1 *For graph drawing representing graph-theoretic distances it is most appropriate to model this representation explicitly in the objective function.*

Given their objectives, both classical and distance scaling should represent graph-theoretic distances well in a geometric layout, and thus be useful for graph drawing. Because of the more direct influence on the objective function and a concave weighting of distance representation errors, it seems plausible that distance scaling would be the more suitable variant for graph drawing. While it is almost commonplace that classical scaling is better at representing global structure whereas distance scaling is better at representing fine details (Buja and Swayne, 2002), we do not know of any systematic evaluation. We therefore provide experimental evidence for the following.

Hypothesis 2 *Distance scaling compares favorably with classical scaling in terms of layout quality, because local details are represented better.*

In our experience, based on many conversations with implementors and users of graph drawing systems, a main reservation against distance scaling is its assumed non-scalability, due to a multitude of local minima the and high computational demand. The next two hypotheses focus on how to ensure that the layouts

produced by implementations of distance scaling are actually those supporting H1.

Hypothesis 3 *Distance scaling is susceptible to poor local minima, because it is highly dependent on the initial layout.*

Hypothesis 4 *Classical scaling provides excellent initial layouts for distance scaling, because the better representation of large distances helps to avoid poor local minima.*

If H4 holds, we have complicated matters even more, because two demanding problems have to be solved rather than one. The final two hypotheses therefore regard the possibility of computing the initial and final layout efficiently.

Hypothesis 5 *Classical scaling layouts of very large graphs can be approximated efficiently using PivotMDS.*

Hypothesis 6 *Distance scaling is practical even on very large graphs.*

5.2 Experimental Design

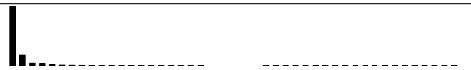
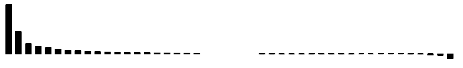
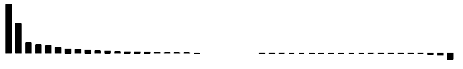
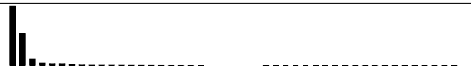
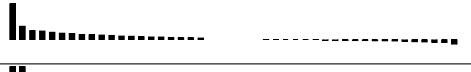
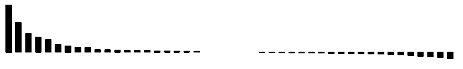
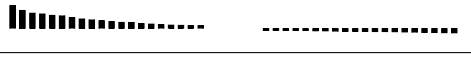
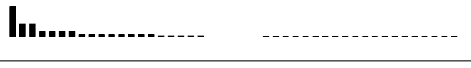
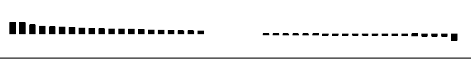
Data The experiments were run on a set of test graphs described in Table 5.1. The graphs were selected large enough to allow for extrapolation of the results to very large graphs, but also small enough to allow for the exact computation of the stress (4.4) in a very large number of experiments.

Note that the eigenvalues of the matrices B associated with each graph indicate the intrinsic dimensionality of the original distances d_{ij}. If, say, two dimensions suffice to reconstruct all the d_{ij}'s exactly, such that the strain criterion is zero, then $\lambda_1 \geq \lambda_2 > \lambda_3 = \cdots = \lambda_n = 0$, and inversely, few large and many (near-) zero eigenvalues indicate the existence of a good low-dimensional layout; see also Figure 3.2.

Environment We implemented all MDS algorithms and speed-up techniques ourselves to avoid bias due to coding, system, or timing. The algorithms were implemented in Java using Sun's SDK 1.6.0 and the yFiles 2.5.0.1 graph library.[1] All experiments were run on a standard 1.4 GHz Compaq NX 7000 notebook with 512 MB of RAM, using Windows XP Service Pack 2.

[1] `http://www.yworks.com`

Table 5.1: Test set of graphs used in Experiments 1–3.

name	n	m	description	diam(G)	spectrum extrema
516	516	729	finite element mesh (Walshaw, 2000)	61	
1138bus	1138	1458	power systems network (Boisvert et al., 1997)	31	
qh882	882	2856	Hydro-Quebec power systems' small-signal model (Boisvert et al., 1997)	31	
plat1919	1919	15240	finite-difference model of shallow wave equations in oceans (Duff et al., 1989)	43	
esslingen1	2075	4769	social network in the city of Esslingen in the 19th century (Lipp, 2000)	15	
sw0	500	1500	cycle in which each node is adjacent to its 3 left and right neighbors	84	
sw002	500	1500	graph sw0, but each edge is rediretced randomly with probability 0.02	27	
sw01	500	1500	graph sw0, but each edge is rediretced randomly with probability 0.1	10	
btree	1023	1022	complete binary tree of height 10	18	
prot1	3025	3629	largest component of a protein interaction network (Ito et al., 2000)	27	

Implementation A simple and convenient way of implementing classical scaling is by constructing the matrix B in (3.4) and computing its two extremal eigenvalues λ_1, λ_2 and eigenvectors v_1, v_2 by power iteration; see Section 3.1.4.

The problem of drawing graphs with fixed edge lengths is $\mathcal{NP}$-hard in general (Eades and Wormald, 1990), and for distance scaling no analytic solution is known, so layouts have to be computed iteratively. In the original proposal by Kruskal (1964), stress is evaluated for the current positions, and new positions are computed by gradient descent; this is also done by Kamada and Kawai (1989); McGee (1966); Sammon (1969) with gradient terms specific to the weights w_{ij}. These approaches were superseded by majorization (de Leeuw, 1977), which generates a sequence of layouts with decreasing stress and can handle arbitrary weights $w_{ij} \geq 0$. In our experiments we use a local iteration with node-by-node updates; see Section 4.1.3.

5.3 Experiments

The first experiments provides evidence for which method yields better layouts in principle (disregarding efficiency, ease of implementation, reliability, etc.), when graph-theoretic distances are to be represented by Euclidean distances. We use the following shorthand notation for the involved algorithms:

- random: nodes are placed randomly with uniform probability in a unit square;
- fm3: the fast multipole multilevel method of Hachul and Jünger (2006) using a combination of a multilevel scheme and fast repulsion force computation;
- grip: a force-directed layout method using a multilevel refinement hierarchy (Gajer and Kobourov, 2001);
- hde: the high dimensional embedder of Harel and Koren (2002b) with 50 pivots, see Section 3.4;
- cmds: classical scaling, see Section 3.1.

Experiment 1 (Layout approach) *All test graphs are laid out with cmds, distance scaling with unweighted and weighted stress, fm3, hde, and grip.*

For convenience, most implementations of iterative layout algorithms start from a random initial configuration. It is, however, widely known that smart initialization is preferable (Malone et al., 2002). We here compare different

initialization strategies for distance scaling and evaluate the resulting stress. Before the iteration all initial solutions X are scaled such that $\sum_{i,j} \|x_i - x_j\| = \sum_{i,j} d_{ij}$.

Experiment 2 (Distance scaling and initialization) *All test graphs are laid out using each of the following layout algorithms:* random, fm3, hde, grip, cmds, *and then minimizing weighted stress using local iteration.*

Classical scaling has running time at least quadratic in the number of nodes n for constructing distance matrix $D \in \mathbb{R}^{n \times n}$ and decomposing the derived matrix $B \in \mathbb{R}^{n \times n}$. Quick estimates for the eigenvectors v_1, v_2 corresponding to λ_1, λ_2 are obtained by using only parts of D by selecting a subset $P \subset V$ of $k \ll n$ *pivot* or *landmark* nodes and taking only $k \cdot n$ rather than n^2 distances into account. Once P is constructed, two approaches for this are considered:

- Pivot MDS (Brandes and Pich, 2007) uses the singular value decomposition of a rectangular matrix: Let $D_P \in \mathbb{R}^{n \times k}$ be the matrix of k columns of distances from nodes in P, e.g. in k breadth-first searches. Then the right singular vectors u_1, u_2 of $C = -\frac{1}{2} J_n D_P^{(2)} J_k$ are estimates for the eigenvectors v_1, v_2 of $B = -\frac{1}{2} J_n D^{(2)} J_n$; see Subsection 3.3.2.
- Landmark MDS (de Silva and Tenenbaum, 2004) places nodes in P by classical MDS. Then each node in $V \setminus P$ is placed based on its k distances to nodes in P; see Subsection 3.3.3.

The k pivots should be well-scattered over the graph; intuitively, this is to represent as much of the full distance information D as possible. Assuming that P contains $j - 1$ selected nodes, our strategies to determine the jth pivot are

- maxmin: $\operatorname{argmax}_{v \in V \setminus P} \min_{p \in P} d_{vp}$, the node farthest from P;
- random: with uniform probability, from P;
- mixed: with maxmin, if k is even, with random otherwise;

combining them with the two estimation approaches above leads to six strategies.

Let $X, Y \in \mathbb{R}^{n \times 2}$ be the estimate and the actual solution, each centered at the origin. To find out how similar X is to Y we use the *Procrustes statistic*

$$R^2 = 1 - \left(\operatorname{tr}(X^T Y Y^T X)^{1/2}\right)^2 / \left(\operatorname{tr}\left(X^T X\right) \cdot \operatorname{tr}\left(Y^T Y\right)\right) \tag{5.1}$$

minimized by the *Procrustes rotation* $P \in \mathbb{R}^{2 \times 2}$ (Sibson, 1978)

$$P = \frac{\operatorname{tr}(X^T Y Y^T X)^{1/2}}{\operatorname{tr}\left(X^T X\right)} \left(X^T Y Y^T X\right)^{1/2} \left(Y^T X\right)^{-1} \in \mathbb{R}^{2 \times 2}\,, \tag{5.2}$$

which, applied to each row in X, optimally dilates, scales, rotates, and reflects X to fit Y. It can be shown that $0 \leq R^2 \leq 1$; if $R^2 = 0$, X and Y can be perfectly matched, if $R^2 = 1$, they cannot be matched by any $P \in \mathbb{R}^{2\times 2}$ at all.

Experiment 3 (Approximating classical scaling) *For each test graph, classical scaling is approximated using 6 strategies* {maxmin, random, mixed} × {landmark, pivot}*, and compared to the exact solutions using the Procrustes statistic.*

Experiments 2 and 3 were repeated 25 times, and to control for biases due to the internal representation of graphs and matrices, we used as many instances of each graph, each with randomly permuted vertices and edges.

Distance scaling by stress minimization is mostly used for improving the representation of local details; setting $w_{ij} = d_{ij}^{-2}$ assigns large weight to the representation of small distances and vice versa. Initializing distance scaling with cmds, we hope that large distances are fitted well; the subsequent fitting of smaller distances and local details is achieved by discarding the large distances from the stress term to be minimized, which we dub *sparse stress*

$$\sigma_S(X) = \sum_{\{i,j\}\in S} w_{ij}(d_{ij} - \|x_i - x_j\|)^2 \ , \tag{5.3}$$

where $S \subseteq V \times V$ is a set of node pairs involved in the iteration, with $|S| \in \mathcal{O}(n)$. In our experiments we use *local neighborhoods* obtained by terminating the breadth-first searches once k nodes are visited.

Experiment 4 (Sparse stress minimization) *For each of the test graphs the initial classical scaling configuration is subjected to sparse stress minimization using only local neighborhoods.*

We use another collection of larger graphs to examine the scalability of initialization and sparse stress minimization. Unlike the test graphs used earlier, their size prohibits methods using the full square matrices. The results are assessed visually with respect to the information known a priori.

Experiment 5 (Very large graphs) *Large graphs are laid out first using an approximation to classical scaling and then sparse stress minimization.*

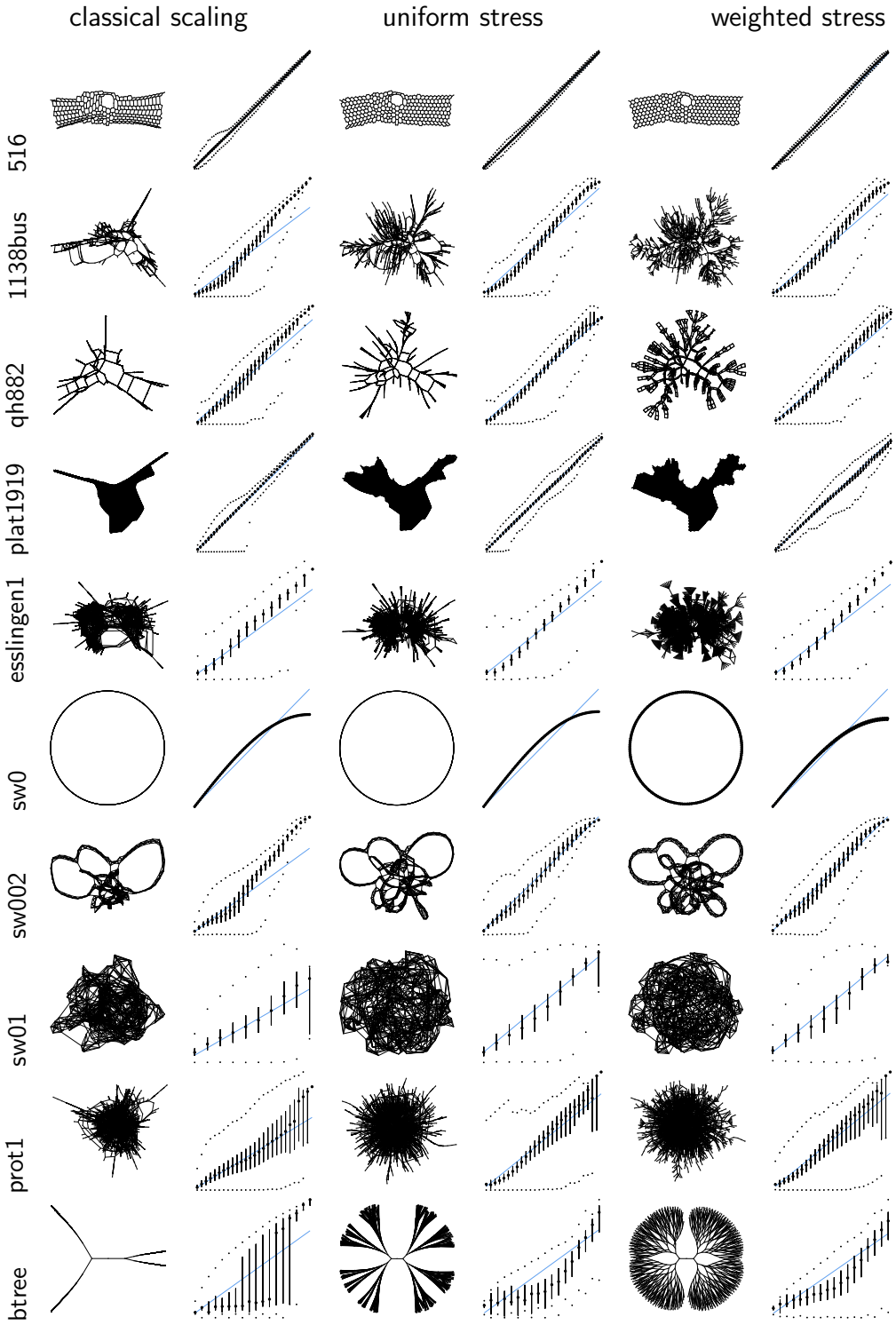


Figure 5.1: Drawings of the test graphs made with classical scaling, distance scaling ($w_{ij} = 1$), and distance scaling ($w_{ij} = d_{ij}^{-2}$), and quartile plots of d_{ij} (abscissa) vs. $\|x_i - x_j\|$. Large dots indicate the median, small dots minimum and maximum, and black lines the range of the two middle quartiles (25–75 per cent). The thin blue line with slope 1 is a visual aid.

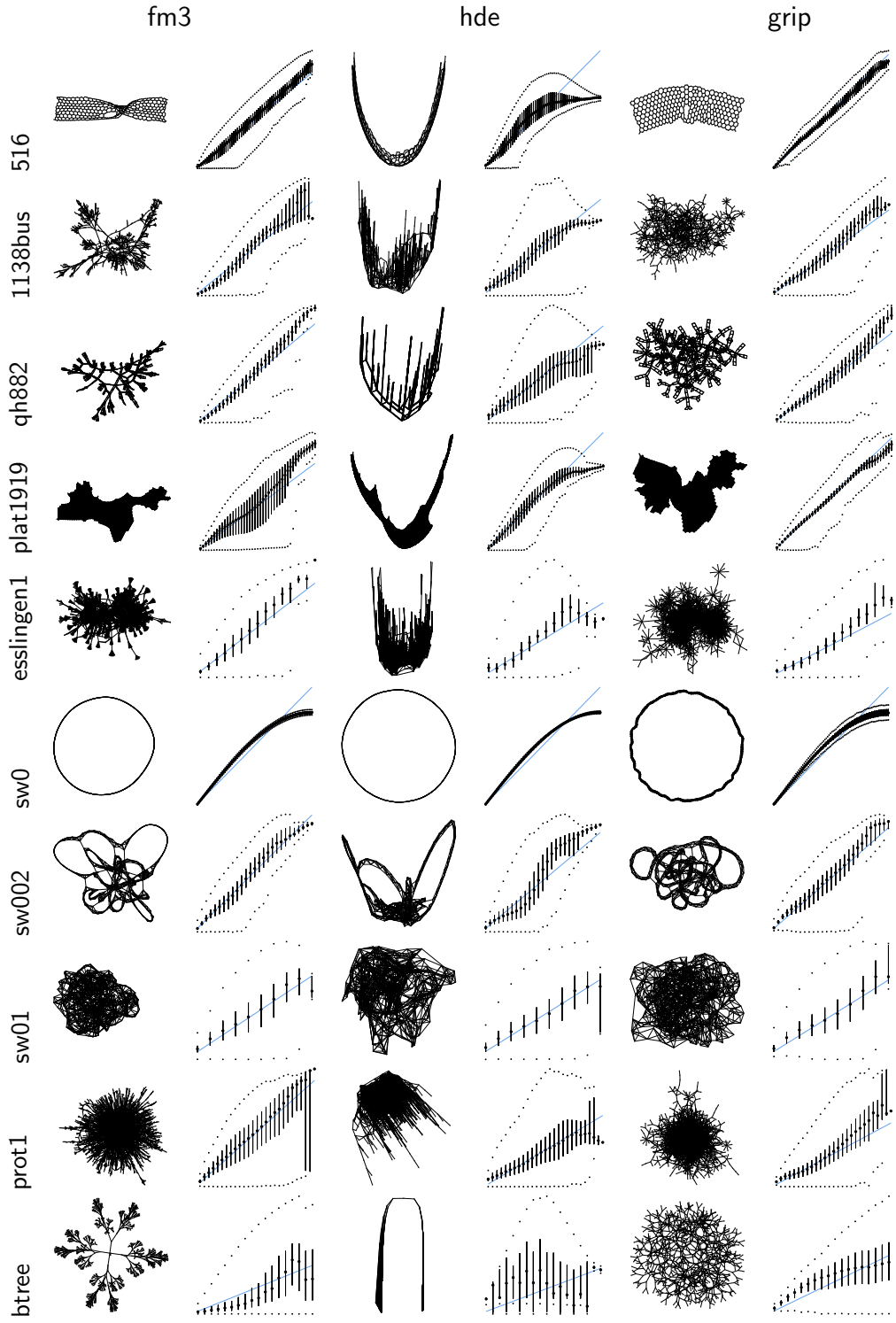


Figure 5.2: Drawings of the test graphs made with FM^3, high-dimensional embedding, and GRIP, and quartile plots of d_{ij} (abscissa) vs. $\|x_i - x_j\|$. Large dots indicate the median, small dots minimum and maximum, and black lines the range of the two middle quartiles (25–75 per cent). The thin blue line with slope 1 is a visual aid.

5.4 Results

Layout Quality. To assess layout quality both visually and quantitatively, aligned layouts and the distributions of layout distances are shown in Figures 5.1 and 5.2 for each of the possible distance values between pairs of vertices, i.e. for values ranging from 1 to the diameter of the respective graph. The classical scaling layouts were generated with random initial positions and used as initial configurations for distance scaling. Initialization is further studied in Experiment 2.

The drawings for graphs qh882, 1138bus seem to confirm H1 and H2; using weights $w_{ij} = d_{ij}^{-2}$ helps to display local structures hidden by classical scaling or unweighted distance scaling. For regular structures 516, plat1919, sw0, distance scaling does not improve the quality of local representation. In a few cases classical scaling represents the overall structure better, such as the known clustering of esslingen1 in two densely connected parts.

In general, *H1 and H2 can be accepted* at least for graphs for which graph-theoretic distance is well representable in low dimensions. However, none of the MDS variants seems to be capable of representing both smaller and larger distances for small diameter graphs and other special types of graphs like btree. In such cases the MDS objective functions for distance representations is not always useful as an aesthetic criterion; see Section 5.5 for a discussion.

Initialization. For independence of graph size and distances we divide the stress by $\sum_{i,j} w_{ij} d_{ij}^2$, which puts all stress values on the same scale and allows for comparison between stress computations even for different graphs. We have carried out the iterative majorization process 25 times for each graph (with permuted edge list) and for each of the five initial placements.

The results of Experiment 2 are displayed in Figure 5.3, which shows stress values over the majorization process for distance scaling, with weights $w_{ij} = d_{ij}^{-2}$ For almost all graphs we have tested, basically the same ranking resulted, with random being worst, followed by fm3, grip, hde. Initially, cmds solutions tend to have higher values, but overtakes the other initializations after some iterations.

All experiments indicate that *H3 is valid for all types of graphs.* Since large distances and thus global structures are represented well, classical scaling gives excellent initial configurations for distance scaling.

The bandwidth of stress values we observed for cmds-initialized layouts was almost always negligible, whereas stress values vary largely for all other methods in the 25 runs. Classical scaling gives reproducible initial configurations throughout, which are also robust against permutation of the input. All these observations *support H4.* Interestingly, btree is the only graph for which classical scaling resulted in some variation; we attribute this to the multiple occurrence of equal eigenvalues of matrix B (see Figure 3.2).

Scalability. We computed estimates for the solution to classical scaling for all graphs, again in 25 runs with random node permutations. In each run, three sets of pivots were grown from $k = 3$ to 120 (following maxmin, random, and mixed) and used for Pivot MDS and Landmark MDS. The plots for the median values of three selected graphs are shown in Figure 5.5.

For graphs with a regular structure, like sw0, 516, the pivoting strategy is not crucial. In all other cases Pivot MDS is superior to Landmark MDS, regardless of the pivoting strategy. For Pivot MDS, the maxmin strategy performs better than random and slightly better than mixed. The corresponding plots seem to converge to zero faster and more smoothly than those for Landmark MDS. Once again, graph btree seems to be different from the others; estimating the full classical scaling solution appears to be unstable, no matter what pivoting strategy is used. Our observations indicate that *H5 is valid*.

We have conducted further experiments considering scalability. One suite of experiments applies Pivot MDS to graphs with millions of nodes; we have observed that even those huge graphs, for which the full classical scaling is impractical, are laid out well with it, provided that two dimensions suffice, and, conversely, that increasing the number of pivots does not improve layout quality if the graph is of higher intrinsic dimensionality; see also Section 5.2.

Another suite of experiments indicates that, technically, stress minimization scales even to very large graphs, but that *H6 is valid only with the limitation* that an appropriate sparsification scheme must be available.

5.5 Conclusion

We have studied different graph-layout approaches that aim at representing graph-theoretic distances by Euclidean distances. Our experiments suggest that minimization of weighted stress, an objective function that models the desired aesthetic properties explicitly, is to be preferred over force-directed placement. The recommended method for weighted stress minimization is to initialize with a fast approximation of classical scaling (Brandes and Pich, 2007) and subsequent iterative improvement using localized stress reduction (Gansner et al., 2004). Both phases are easy to implement, but the second can be time-consuming. Approximation via sparse stress makes the algorithm scale to very large graphs, but further research on reliable sparsification schemes is needed.

The distance-based approach yields poor results on certain classes of graphs, which include small worlds and other graphs with many shortcuts or low diameter, and scale-free graphs with highly skewed degree distributions, large 1-shells, or other forms of structural imbalance. Some success has been obtained with stress weighting schemes based on graph invariants, but good characterizations

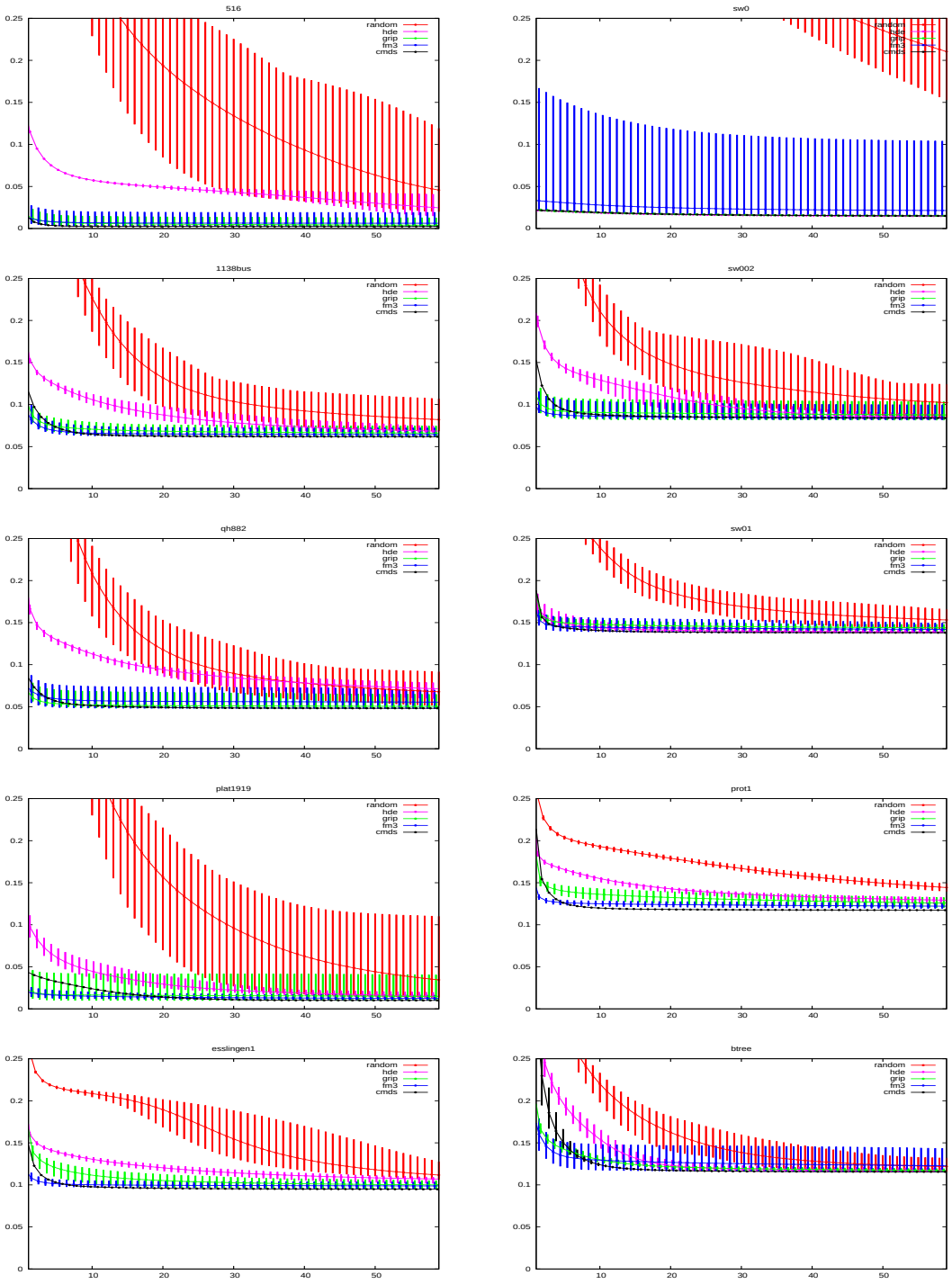


Figure 5.3: Upper row: The majorization process with different initializations random, fm3, hde, grip, cmds after $0, 30, 60$ iterations. Lower row: Number of iterations vs. stress. The bars indicate the range of values, the dots the median value, in 25 runs.

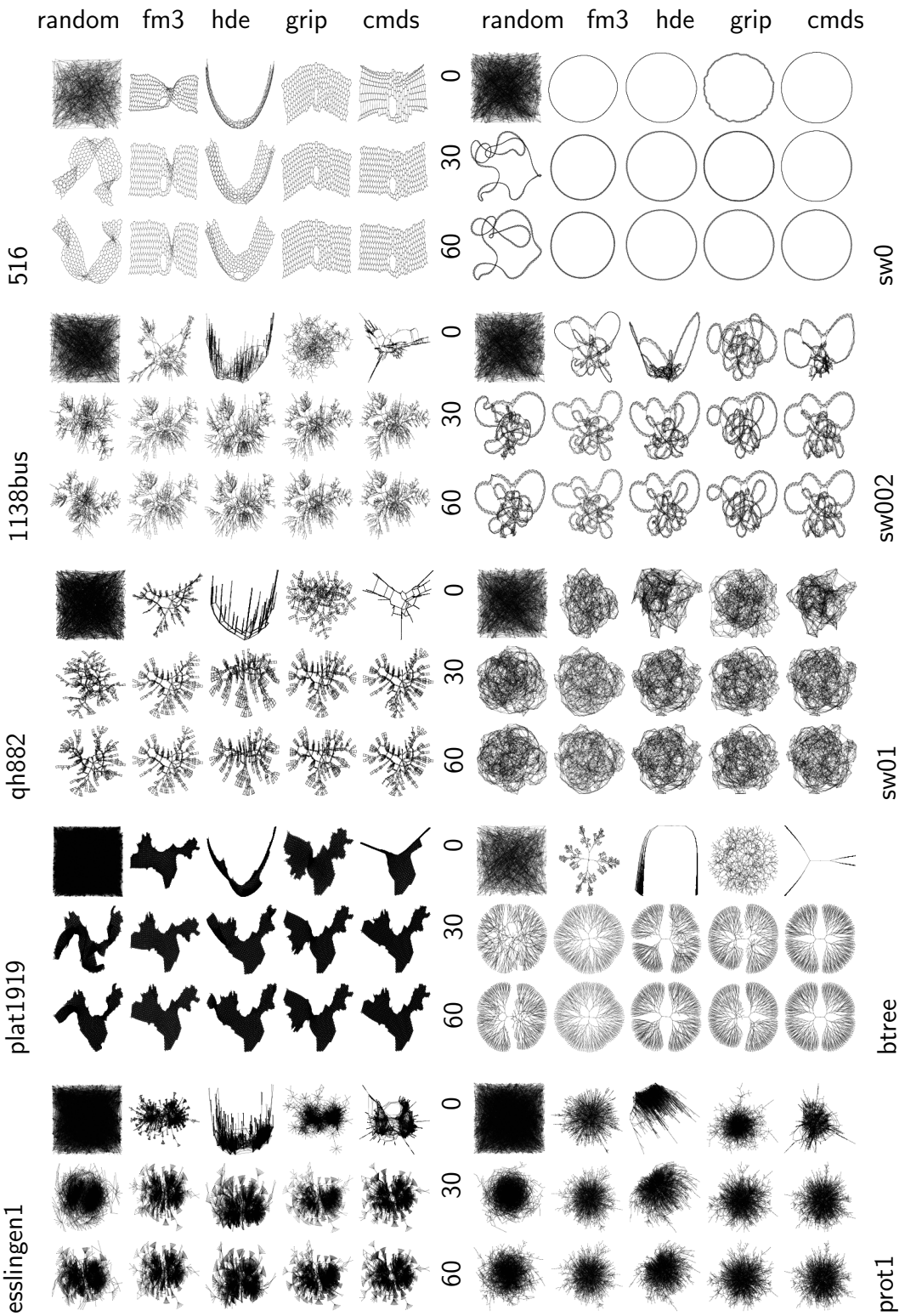


Figure 5.4: Drawings of all test graphs, with the five initial configurations (each block, from left to right: random, fm3, hde, grip, cmds). The rows are layouts after 0, 30, 60 iterations of distance scaling, $w_{ij} = d_{ij}^{-2}$.

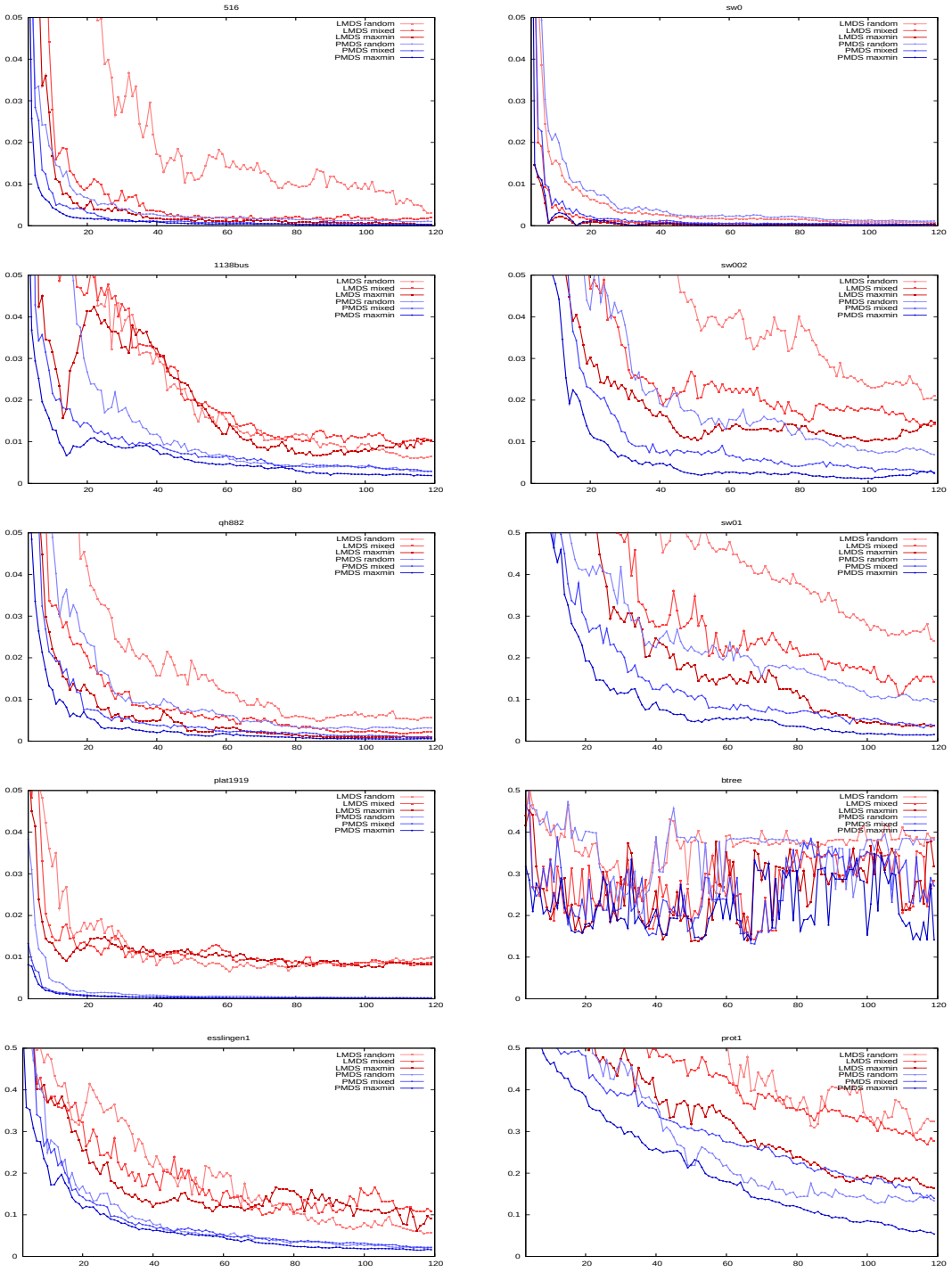


Figure 5.5: Procrustes statistics measuring how well Pivot MDS (red) or Landmark MDS (blue) estimate the exact solution of classical scaling. Plotted are the median values of 25 runs with different node permutations, for $k \in \{3, \ldots, 120\}$ pivots.

of problematic graphs are missing and matching layout algorithms need to be developed further.

Using a hypotheses-based experimental design, we hope to foster clarity and reproducibility of our results, and to contribute to experimental evaluation of graph drawing algorithms in general.

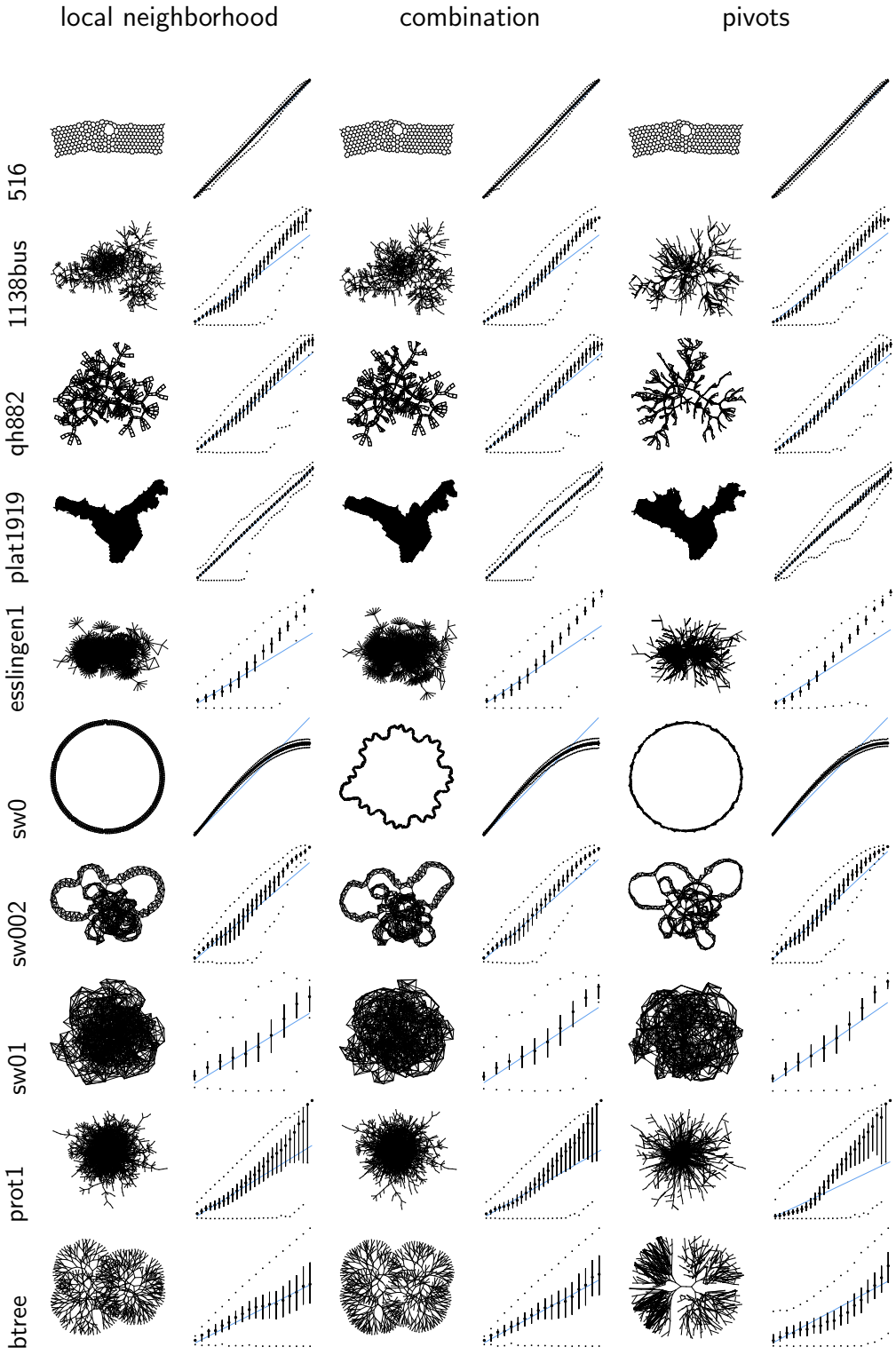


Figure 5.6: Drawings of all test graphs after minimization of sparse stress (50 iterations), using different versions of sparse stress. Left column: Local neighborhoods with at most 100 nearest neighbors. Right column: Distances to 100 pivots selected with the minmax strategy. Middle column: The combination of left and right.

Chapter 6

Skew-symmetric Scaling

The use of multidimensional scaling methods representing distances is limited to the visualization of undirected graphs, since, in Euclidean spaces, proximity and distance are notions which are inherently symmetric; it is not straightforward how to adapt this to directed graphs, since the associated matrices are generally not symmetric. There are some strategies of modifying the asymmetric input data such that MDS for symmetric data can be applied.

When the direction of edges is not crucial for the structure of the graph to be visualized, the directional information may simply be ignored, and MDS is applied to the underlying undirected graph, whose adjacency and distance matrices are symmetric. Another way of obtaining a symmetric distance matrix is to set the distance between nodes i and j to be the average distance $\hat{d}_{ij} = (d_{ij} + d_{ji})/2 = \hat{d}_{ji}$ in both directions.

In the first case, where edge directions are ignored, the asymmetry is completely removed by symmetrizing locally; in the second case, the asymmetry is removed globally. While in most cases these approaches are sufficient for drawing the original directed graph, neither of them takes the inherent asymmetry of directed graphs, which may be interpreted as dominance, direction, or influence, fully into account.

This chapter discusses an MDS method for the explicit analysis of asymmetry data and its application to directed graphs. While, algebraically, the method is closely related to classical scaling (see Chapter 3), the information to be visualized and the interpretation of the visualization are quite different. The asymmetric data is partitioned into a symmetric and a purely asymmetric component, both of which can be analyzed and visualized separately. The purely asymmetric component is captured by *skew-symmetric* matrices, whose analysis and the corresponding visualization method is termed *skew-symmetric scaling*; in the literature, it is also called *canonical analysis of skew-symmetry.*

6.1 Skew-Symmetry

6.1.1 Skew-Symmetric Matrices

The following result is attributed to folklore. It states that any real square matrix $A = (d_{ij}) \in \mathbb{R}^{n\times n}$ can be uniquely partitioned into

$$A = R + S,$$

where $R = (r_{ij}) \in \mathbb{R}^{n\times n}$ is symmetric, $R = R^T$, and $S = (s_{ij}) \in \mathbb{R}^{n\times n}$ is *skew-symmetric*, $S = -S^T$. This partition is obtained by setting

$$R = \frac{1}{2}(A + A^T), \quad S = \frac{1}{2}(A - A^T).$$

The entry $r_{ij} = (a_{ij} + a_{ji})/2 = r_{ji}$ of R represents a symmetric proximity of objects i and j, which is defined as the average dissimilarity in both directions. In the skew-symmetric component, the entry $s_{ij} = (a_{ij} - a_{ji})/2 = -s_{ji}$ of matrix S is usually interpreted as a dominance relation between i and j. A positive sign of s_{ij} indicates that i *dominates* j and that j is dominated by i, a negative sign the opposite.

In a similar fashion, the corresponding sum of squared entries can also be partitioned by

$$\begin{aligned}
\sum_{i,j} a_{ij}^2 &= \sum_{i,j} (r_{ij} + s_{ij})^2 \\
&= \sum_{i,j} (r_{ij}^2 + 2r_{ij}s_{ij} + s_{ij}^2) \\
&= \sum_{i,j} r_{ij}^2 + 2\sum_{i,j} r_{ij}s_{ij} + \sum_{i,j} s_{ij}^2 \\
&= \sum_{i,j} r_{ij}^2 + \frac{1}{2}\left(\sum_{i,j} a_{ij}^2 - \sum_{i,j} a_{ji}^2 + \sum_{i,j} a_{ij}a_{ji} - \sum_{i,j} a_{ji}a_{ij}\right) + \sum_{i,j} s_{ij}^2 \\
&= \sum_{i,j} r_{ij}^2 + \sum_{i,j} s_{ij}^2 \ ,
\end{aligned}$$

which indicates that it makes sense to treat the two components with a least-squares analysis separately.

The *asymmetricity* of a matrix is measured as the relative share of the sum of squared entries

$$\sum_{i,j} s_{ij}^2 / \sum_{i,j} a_{ij}^2 \tag{6.1}$$

A detailed account of asymmetric proximities and their analysis can be found in Zielman and Heiser (1996). Harshman and Lundy (1990) pay special attention

to the interpretation of the solutions. The combination of visual representations for both the symmetric and the skew-symmetric part is discussed by Gower and Digby (1981).

6.1.2 Decomposition

The dominance information contained in the skew-symmetric matrix can be visualized using a spectral decomposition similar to the one in classical scaling in Chapter 3. While in classical scaling a symmetric matrix is decomposed into a sum of symmetric rank-one matrices, in skew-symmetric scaling a skew-symmetric matrix is decomposed into a sum of skew-symmetric rank-two matrices. The spectral decomposition of a skew-symmetric matrix S has the form

$$S = U\Phi U^T \ , \tag{6.2}$$

where $U \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose columns are real eigenvectors $u_1, \ldots, u_n \in \mathbb{R}^n$, without loss of generality of unit length, $\|u_j\| = 1$ for all $j \in \{1, \ldots, n\}$, and $\Phi \in \mathbb{C}^{n\times n}$ is a diagonal matrix of complex eigenvalues.

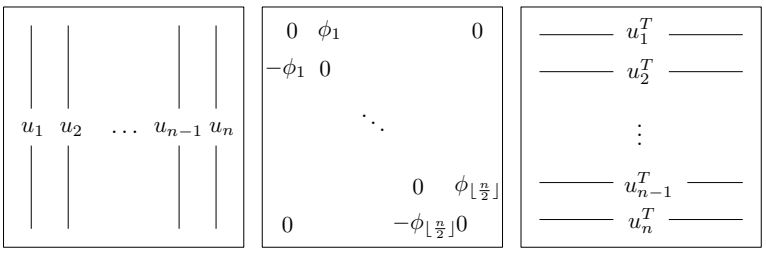
Due to the skew-symmetry, the eigenvalues are purely imaginary and occur in pairs

$$\pm\phi_1 \cdot \sqrt{-1}, \pm\phi_2 \cdot \sqrt{-1}, \ldots, \pm\phi_{\lfloor n/2 \rfloor} \cdot \sqrt{-1}$$

with an additional zero eigenvalue if n is odd.

We will refer to a pair of eigenvalues $\pm\phi_i \cdot \sqrt{-1}$ with the same imaginary part as *the eigenvalue* ϕ_i. Without loss of generality, the eigenvalues in Φ are ordered non-increasingly by their absolute magnitude, $\phi_1 \geq \cdots \geq \phi_{\lfloor n/2 \rfloor} \geq 0$. With every eigenvalue ϕ_i, a pair of eigenvectors u_{2i-1}, u_{2i}, $1 \leq i \leq \lfloor n/2 \rfloor$ is associated, spanning a two-dimensional space, called *(i-th) bimension.*

Through orthogonal transformation, (6.2) can be brought into a slightly different form known as the *Gower decomposition* (Gower, 1977), which reads



(6.3)

and allows S to be written as a sum of $\lfloor n/2 \rfloor$ elementary rank-2 matrices

$$S = \sum_{i=1}^{\lfloor n/2 \rfloor} \phi_i \left(u_{2i-1} u_{2i}^T - u_{2i} u_{2i-1}^T \right), \tag{6.4}$$

all of which are skew-symmetric. Thus, each of these $\lfloor n/2 \rfloor$ components explains a share of the skew-symmetry information contained in S; the magnitude of the eigenvalue ϕ_i tells how large the share of bimension i is. Note that a pair of eigenvectors u_{2i-1}, u_{2i} can be replaced by any orthogonal pair of vectors spanning the same two-dimensional space.

Let $\hat{S} = (\hat{s}_{ij})$ denote the partial sum $\sum_{i=1}^{r} \phi_i (u_{2i-1} u_{2i}^T - u_{2i} u_{2i-1}^T)$ of the r first terms in (6.4). It can be shown that this skew-symmetric matrix is an optimal least-squares approximation to S, minimizing the residual error

$$\left\| S - \hat{S} \right\|^2 = \mathrm{tr}\left((S - \hat{S})(S - \hat{S})^T \right) = \sum_{i,j} (s_{ij} - \hat{s}_{ij})^2 \tag{6.5}$$

among all skew-symmetric matrices of rank $2r$ (Gower and Constantine, 1978). If n is odd, the singleton zero eigenvalue does not contribute to the sum in (6.4).

6.1.3 Eigenvalues

For a skew-symmetric matrix S, the identity $SS^T = -S^2$ is very useful. The two eigenpairs $-\sqrt{-1}\phi_i, u_{2i-1}$ and $+\sqrt{-1}\phi_i, u_{2i}$ of the i-th bimension and the corresponding eigenvectors of SS^T, i.e., the left singular vectors of S, are related by the equations

$$\begin{aligned} S u_{2i-1} &= -\phi_i \sqrt{-1} u_{2i-1} \\ S u_{2i} &= +\phi_i \sqrt{-1} u_{2i} \end{aligned}$$

and

$$\begin{aligned} SS^T u_{2i-1} &= -S^2 u_{2i-1} &&= -(-\phi_i \sqrt{-1})^2 u_{2i-1} &&= \phi_i^2 u_{2i-1} \\ SS^T u_{2i} &= -S^2 u_{2i} &&= -(+\phi_i \sqrt{-1})^2 u_{2i} &&= \phi_i^2 u_{2i}, \end{aligned}$$

where it is used that eigenvalues of S^2 are the squared eigenvalues of S. We exploit that SS^T is symmetric for computing the largest eigenvectors and eigenvalues of S in Subsection 6.1.5.

Using the equality of the traces and the eigenvalue sums of the symmetric matrices SS^T and $\hat{S}\hat{S}^T$, the residual error criterion (6.5) can also be formulated in terms of eigenvalues as

$$\mathrm{tr}\left((S - \hat{S})(S - \hat{S})^T \right) = 2 \sum_{i=1}^{\lfloor n/2 \rfloor} \phi_i^2 - 2 \sum_{i=1}^{r} \phi_i^2 = 2 \sum_{i=r+1}^{\lfloor n/2 \rfloor} \phi_i^2$$

which is obviously minimal when $\sum_{i=1}^{r} \phi_i^2$ is maximal; this is reminiscent of the strain criterion (3.11) in classical scaling. The value of the fraction

$$\sum_{i=1}^{r} \phi_i^2 \Big/ \sum_{i,j} s_{ij}^2 \tag{6.6}$$

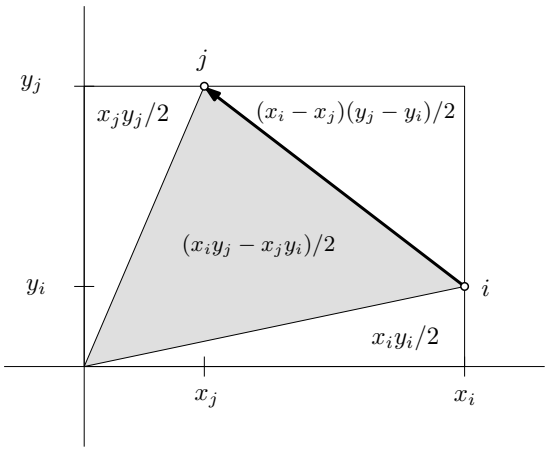


Figure 6.1: The signed area of the triangle subtended by the origin and the positions (x_i, y_i) and (x_j, y_j) represents the skew-symmetry between i and j.

is between zero and one and is the relative share of the first r bimensions of the entire skew-symmetry. Harshman and Lundy (1990) give some details about the use and combination of a set of bimensions in preference data and their interpretation.

The bimension associated with ϕ_1 is particularly interesting because it directly yields an optimal two-dimensional representation of the skew-symmetry, i.e., the best skew-symmetric rank-2 approximation to K. Coordinates for every object are simply obtained by setting

$$x = \sqrt{\phi_1}u_1, y = \sqrt{\phi_1}u_2 \tag{6.7}$$

and in other bimensions analogously.

6.1.4 Interpretation

In a bimension spanned by two vectors $x = [x_1, \ldots, x_n]^T$, $y = [y_1, \ldots, y_n]^T$ resulting from (6.7), the quantity s_{ij}, which expresses the particular skew-symmetry among i and j, is fitted by

$$s_{ij} \approx x_jy_i - x_iy_j, \tag{6.8}$$

which is proportional to the *signed area* of the triangle formed by the origin and the positions (x_i, y_i) and (x_j, y_j) of the objects i and j, since

$$x_iy_j - (x_jy_j/2 + x_iy_i/2 + (x_i - x_j)(y_j - y_i)/2) = (x_iy_j - x_jy_i)/2$$

as is shown in Figure 6.1.

Unlike other multidimensional scaling methods, the interpretation of the drawing is not based on the Euclidean distance between positions of nodes in the drawing, which is a representation relying on the symmetry of distances; rather, the skew-symmetry is represented in terms of the orientation and the size of triangle areas for all pairs of nodes.

Since two eigenvectors spanning a bimension share the same eigenvalue, axes are not meaningful, and the configuration may be freely rotated around the center without modifying triangle areas and signs. Furthermore, it is not determined whether the bimension blocks in the block-diagonal matrix Φ in (6.3) are of the form $\left(\begin{smallmatrix} 0 & \phi_i \\ -\phi_i & 0 \end{smallmatrix}\right)$ or $\left(\begin{smallmatrix} 0 & -\phi_i \\ \phi_i & 0 \end{smallmatrix}\right)$, so that the orientation of the configuration is made clockwise or counterclockwise, as desired, by reflecting it around any line passing through the origin.

Another difference between the representation of skew-symmetry and the representation of distances is that the origin is an integral part of the layout because the triangle areas are always defined with respect to it. When two nodes are collinear on a line through the origin, their asymmetry is low because the corresponding triangle areas are zero or very small; note that this also holds for objects on opposite sides of a line through the origin.

The one-bimensional representation is especially useful when the original skew-symmetric matrix is inherently circular; in such cases, the bimensions contain circular triples, as in Figure 6.2(a). No such circular triples are present when all objects in a configuration are on the same side of a line through the origin, as in Figure 6.2(b). In this case, the angular order gives a natural ranking of all objects, which indicates that the objects are intrinsically non-circular at least in this bimension, which is typically the first one.

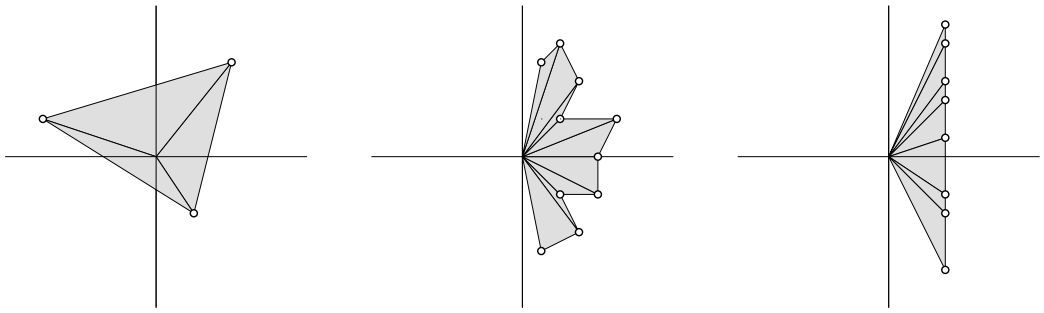
Figure 6.2(c) shows a special case of a non-circular configuration. Such a linear arrangement can be rotated in such a way that $y = [1, \ldots, 1]^T$, and the skew-symmetry between two objects i and j is fitted by

$$S \approx x1_n^T - 1_n x^T$$

and

$$s_{ij} \approx x_i \cdot 1 - 1 \cdot x_j = x_i - x_j.$$

This is the simplest form of skew-symmetry, which allows the triangle areas in this bimension to be interpreted as a signed distance between objects, forming an additive scale. While this model of skew-symmetry is very restricted, it is useful for the interpretation of configurations which are not exactly, but almost linear, or when only a subset of all objects is placed on a line.



(a) a cyclic triple **(b)** no cyclic triples **(c)** linear model

Figure 6.2: Cases of skew-symmetry in a bimension.

6.1.5 Computation

There are some dedicated algorithms for computing the complete decomposition (6.3) of a skew-symmetric matrix (Paardekooper, 1971; Ward and Gray, 1978). When only the largest eigenvalues and their eigenvectors are required, a simple power iteration is sufficient. The entries of these eigenvectors may be directly used as coordinates.

Since $S = -S^T$ and thus $-SS^T = S^2$, the desired singular vectors are conveniently computed using the symmetric matrix SS^T with eigenvalues $\phi_1^2 \geq \phi_2^2 \geq \ldots$ of the imaginary parts of the eigenvalues of S. Pseudocode is given in Algorithm 10.

The decomposition is computed with power iteration, by carrying out the multiplication step

$$x \leftarrow \frac{SS^T x}{\|SS^T y\|}$$

iteratively, as in Algorithm 2 in Section 3.1. Instead of materializing the matrix SS^T, which requires $\mathcal{O}(n^3)$ multiplication operations, it is sufficient to iterate the two steps

$$x \leftarrow \frac{S^T x}{\|S^T x\|}, \quad x \leftarrow \frac{Sx}{\|Sx\|}$$

which requires only $\mathcal{O}(n^2)$ multiplications, assuming a constant number of iterative steps to achieve convergence.

Algorithm 9: Decomposing a skew-symmetric matrix

Input: skew-symmetric matrix $S \in \mathbb{R}^{n \times n}$, bimensionality $b \in \mathbb{N}$
Output: eigenvalues $\phi_1, \ldots, \phi_b \in \mathbb{R}$, eigenvectors $u_1, \ldots, u_{2b} \in \mathbb{R}^n$

$u_1, \ldots, u_{2b} \leftarrow$ random
while *(relative change in any* u_i*)* $> \epsilon$ **do**
  for $i = 1, \ldots, 2b$ **do**
    $u_i \leftarrow \frac{S^T u_i}{\|S^T u_i\|}$
    $u_i \leftarrow \frac{S u_i}{\|S u_i\|}$
    for $j = 1, \ldots, i-1$ **do**
      $u_i \leftarrow u_i - \frac{\langle u_i, u_j \rangle}{\langle u_j, u_j \rangle} u_j$
for $i = 1, \ldots, b$ **do**
  $\phi_i \leftarrow \sqrt{\langle S u_{2i}, S u_{2i} \rangle}$

6.2 Clockwise Drawing of Directed Graphs

Directed graphs are frequently drawn with the desire to have as many edges as possible point in the same direction, say, downwards; it is assumed that there is a general trend or direction of flow in the graph. The most popular and thoroughly researched drawing method is the Sugiyama framework (Sugiyama et al., 1981), which has shown to be very useful for directed graphs with no or only few cycles. After a preprocessing step, in which some edges are temporarily removed or reversed, the graph is acyclic and all edges can be drawn to point in the same direction.

An alternative approach is the minimization of *hierarchy energy* (Carmel et al., 2004). Every edge in a directed graph induces a *target height difference* between the two adjacent nodes; an iterative optimization process tries to find vertical coordinates which attain these height differences as well as possible. Unlike the Sugiyama scheme, the nodes are not assigned discrete levels, but continuous vertical coordinates.

In some applications, however, it is not appropriate to assume that there is some reasonable partial order in the graph. For example, directed cycles may not just be considered as "error", but may represent essential information, which should be highlighted and conveyed adequately in a drawing. Such *recurrent hierarchies* were introduced by Sugiyama et al. (1981), but have gone unnoticed for a long time until recently.

An alternative to the traditional style of hierarchical layouts are *clockwise drawings*; they are visual representations for such cyclic or recurrent hierarchies,

in which the lowest level is considered to be on top of the highest level again. A clockwise drawing has a distinguished center point, and the drawing is read clockwise with respect to this center point. The key task is to find a cyclic ordering in which a node appears after its predecessors and before its successors. Consequently, when an edge is drawn from its source to its target node, it should follow the clockwise orientation. For clockwise drawings, the following criteria should be met as well as possible:

- Edges should run clockwise,
- nodes should be close to their adjacent nodes, and
- nodes should be scattered over the drawing.

Sugiyama and Misue (1995) have introduced a set of modifications of force-directed algorithms to enforce such a cyclic orientation. In this context, they ask: "Can directed graphs with cycles be drawn in a way that it is easy to grasp the global flow of the graphs and the existence of cycles?" They use a concentric force field which rotates around the center and takes edges along, and report about promising experimental results for small example graphs. Their method is quite intuitive and easy to implement, but, as is common to force-directed methods, highly susceptible to local minima, and sensitive to the choice of initial configurations.

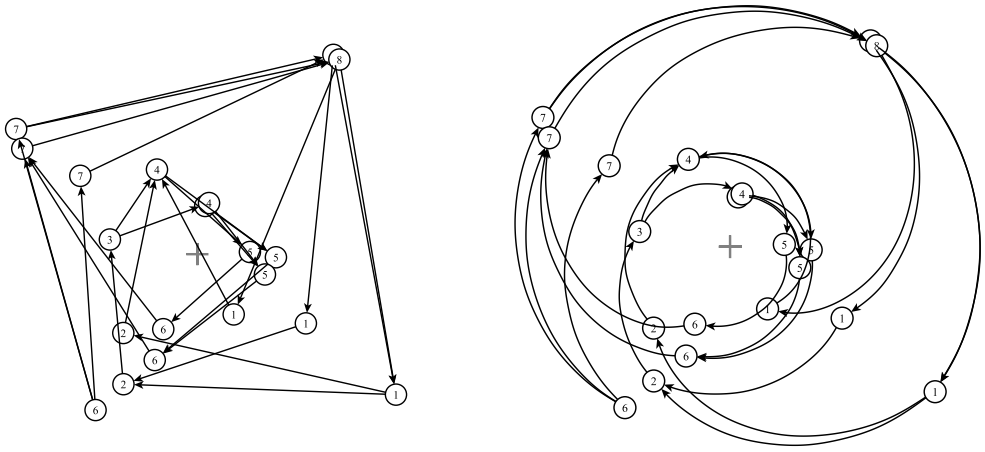
Bachmaier et al. (2009) extend the classic Sugiyama style by a cyclic leveling. The lowest level is considered to be adjacent to the highest level; this modification renders some of the involved optimization problems $\mathcal{NP}$-complete, so that heuristics have to be used. For the combinatorial background of the cyclic arrangement of directed graphs, see also (Ganapathy and Lodha, 2004; Liberatore, 2004).

We describe a novel approach for drawing directed graphs in a cyclic style, which does not require a discrete leveling, and gives direct and optimal solutions. Skew-symmetric scaling is applied to the skew-symmetric component of adjacency matrices. See Figure 6.3 for drawings of an example graph. We give the mathematical background and exploit the sparsity of the adjacency matrix of a graph to obtain a linear-time drawing algorithm.

6.2.1 Skew-Symmetric Adjacency Matrices

In the following it is assumed that there is at most one edge between any two nodes, and that there are no self-edges. The *skew-symmetric adjacency matrix* $S = S(G) = (s_{vw})$ of a directed graph $G = (V, E)$ is derived from its adjacency matrix A by

$$S = A - A^T \tag{6.9}$$



(a) Straight line edges **(b)** Curved edges

Figure 6.3: Clockwise drawings of the graph on the cover page of (Kaufmann and Wagner, 2001). The crosses indicate the location of the origin, relative to which the configuration is oriented. The labels represent the layers in the original drawing.

with entries

$$s_{vw} = \begin{cases} 1 & \text{if } (v,w) \in E, (w,v) \notin E \\ -1 & \text{if } (w,v) \in E, (v,w) \notin E \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

for all $v, w \in V$.

6.2.2 A Two-Dimensional Drawing Method

The skew-symmetric scaling procedure is applied to the skew-symmetric adjacency matrix to obtain two-dimensional drawings, which are oriented either clockwise or counterclockwise, relative to the origin. When node v precedes w in the graph, v should also precede w in the sense of the orientation of the drawing, i.e., edge should (v, w) point forward.

For a directed graph $G = (V, E)$ we use the first bimension of the decomposition (6.3) of $S = S(G)$. Coordinates are simply obtained by setting $x = \sqrt{\phi_1}u_1, y = \sqrt{\phi_1}u_2$ (Gower and Constantine, 1978). These coordinates minimize

$$\sum_{(v,w)\in E} (x_v y_w - x_w y_v - 1)^2 + \sum_{(v,w),(w,v)\notin E} (x_v y_w - x_w y_v)^2 \tag{6.11}$$

among all $x, y \in \mathbb{R}^n$. This criterion tries to represent each directed edge with a unit area triangle in the same direction, and is equivalent to the absolute share contained in the remaining $\lfloor n/2 \rfloor - 1$ bimensions, recast as

$$\left\| S - (xy^T - yx^T) \right\|^2 = \phi_2^2 + \cdots + \phi_{\lfloor n/2 \rfloor}^2. \tag{6.12}$$

To assess how well the skew-symmetry information in $S(G)$ is represented by a bimension, we use that the trace of a symmetric matrix is equal to the sum of its eigenvalues, and hence, that the sum of squared eigenvalues is the trace of the squared matrix. Applying this to the symmetric matrix $S^2 = -(A - A^T)^2 = (A - A^T)(A - A^T)^T$ gives

$$\begin{aligned} 2 \sum_{i=1}^{\lfloor n/2 \rfloor} \phi_i^2 &= \operatorname{tr}(S^2) \\ &= -\operatorname{tr}(A - A^T)(A^T - A) \\ &= -\sum_{v \in V} \sum_{w \in N^-(v) \uplus N^+(v)} -1 \\ &= \sum_{v \in V} (\deg^-(v) + \deg^+(v)) \\ &= 2m \end{aligned}$$

and thus

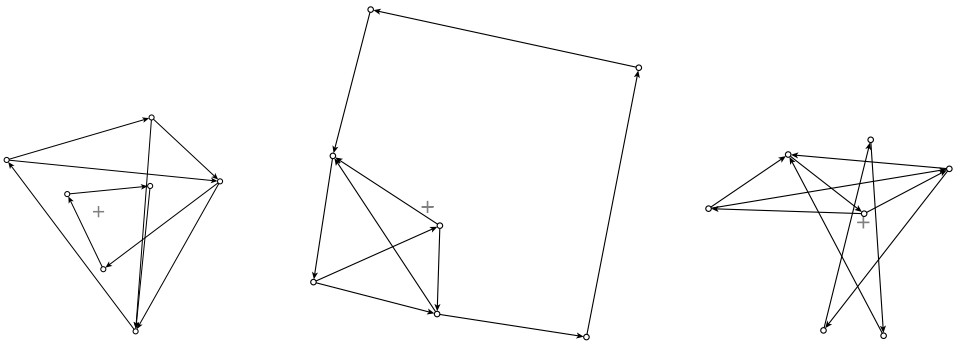
$$\sum_{i=1}^{\lfloor n/2 \rfloor} \phi_i^2 = m, \tag{6.13}$$

so that the number m of edges can be directly interpreted as a measure of skew-symmetry in the graph. The relative share of skew-symmetry explained by the first r bimensions is

$$\frac{\phi_1^2 + \cdots + \phi_r^2}{\phi_1^2 + \cdots + \phi_{\lfloor n/2 \rfloor}^2} = \frac{\phi_1^2 + \cdots + \phi_r^2}{m}, \quad 1 \le r \le \lfloor n/2 \rfloor. \tag{6.14}$$

Although the bimension for the largest eigenvalue is the best one can do with only one bimension, it is also useful to draw a graph in other bimensions. A small graph and the layout in all its bimensions is given in Figure 6.4. The second bimension explains as much as possible of the remaining skew-symmetry after removing the first bimension, and so on.

Even though rarely occurring in practical applications, it should be noted that in the skew-symmetric adjacency matrices of some graphs with a very regular structure, eigenvalues occur with a multiplicity of $k > 1$, i.e., $\phi_i = \phi_{i+1} = \cdots = \phi_{i+k-1}$, for example, in directed cycles of even length. Then, the corresponding k bimensions span a $2k$-dimensional eigenspace, and any two-dimensional subspace



(a) $\phi_1 = 2.65843$ **(b)** $\phi_2 = 1.55086$ **(c)** $\phi_3 = 0.72802$

Figure 6.4: Drawings of an example graph ($n = 7, m = 10$) in all three possible bimensions. Note that $\phi_1^2 + \phi_2^2 + \phi_3^2 = m$, and the bimensions account for about $\phi_1^2/m \approx 70.6\%$, $\phi_2^2/m \approx 24.1\%$ and $\phi_3^2/m \approx 5.3\%$ of the skew-symmetry information.

in this eigenspace is also a bimension for that eigenvalue. Numerically, this problem also occurs at small eigengaps, when consecutive eigenvalues are close together, i.e., $\phi_i/\phi_{i+1} < 1 + \epsilon$ for some small $\epsilon > 0$.

To avoid unnecessary and misleading crossings by straight lines, edges can be routed clockwise by drawing them as curves. A simple way to achieve smooth curves is to use splines or polylines with a reasonable choice of control points, through which the spline curve has to pass. The positions of control points are determined by the *polar coordinates* $(\alpha_v, \rho_v), (\alpha_w, \rho_w)$ of the nodes v, w given by the polar transformation

$$\begin{array}{rclcrcl} \rho_v & = & \sqrt{x_v^2 + y_v^2} & & \alpha_v & = & \operatorname{atan2}(x_v, y_v) \\ \rho_w & = & \sqrt{x_w^2 + y_w^2} & & \alpha_w & = & \operatorname{atan2}(x_w, y_w) \end{array} \tag{6.15}$$

where $\operatorname{atan2}(\cdot,\cdot)\colon \mathbb{R}^2 \to [0, 2\pi]$ denotes the two-argument inverse of the tangent function which is implemented in most modern programming languages. Linear interpolation gives

$$\rho(t) = (1-t)\cdot\rho_v + t\cdot\rho_w \tag{6.16}$$

$$\alpha(t) = (1-t)\cdot\alpha_v + t\cdot\alpha_w \tag{6.17}$$

where $0 \le t \le 1$. When k control points are used, $t \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1\}$. Note that when $|\alpha_v - \alpha_w| > \pi$, the interpolation (6.15) results in the edge (v, w) winding around the center with an angle greater than π; the shorter counterpart of that curve is obtained by adding 2π to the smaller of α_v, α_w.

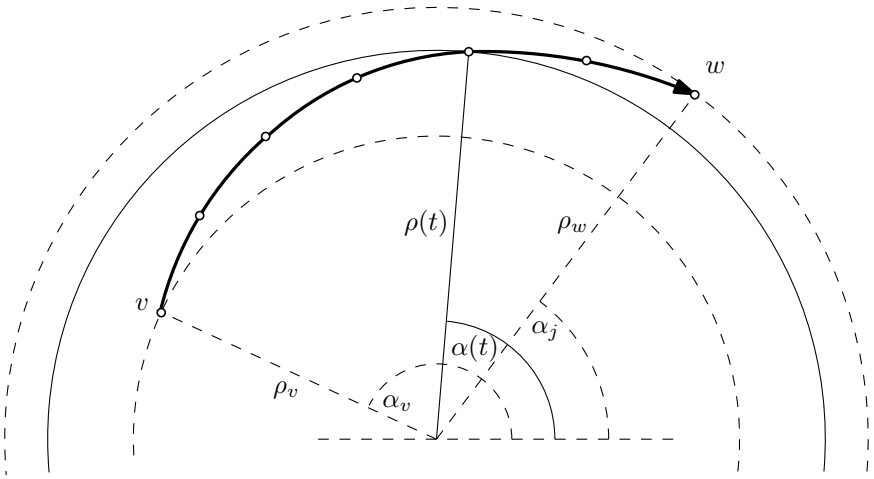


Figure 6.5: Drawing curved edges using splines with control points.

6.2.3 Computation

A straightforward implementation of the power iteration has $\Theta(n^2)$ running time per step. When the graph is sparse, $m \in o(n^2)$, the power iteration can be implemented to run in linear-time, using that S has $\Theta(m)$ non-zero entries, and making the realistic assumption that a constant number of steps is sufficient for the iteration to converge.

The power iteration process is carried out with the symmetric matrix $SS^T = (A - A^T)(A^T - A)$ in the two intermediate steps

$$\hat{x} \leftarrow (A - A^T) \cdot x, \quad \hat{y} \leftarrow (A - A^T) \cdot y \tag{6.18}$$

$$x \leftarrow (A^T - A) \cdot \hat{x}, \quad y \leftarrow (A^T - A) \cdot \hat{y} \tag{6.19}$$

each of which just requires a linear scan over all edges; this is similar to the computation of *hubs and authorities* (Kleinberg, 1999), where the eigenvectors of AA^T and A^TA are required. Pseudo-code for an algorithm with $\mathcal{O}(n + m)$ running time is given in Algorithm 10. Further bimensions are obtained by orthogonalizing the current iterate against all previously computed eigenvectors.

An intuitive interpretation of (6.18) is that on every node clockwise forces are exerted; predecessors exert repulsive forces, successors exert attractive forces. Together with (6.19) and the orthogonalization, the result can be thought of as a rotating force around the origin, where the arrows Figure 6.6 symbolize the movement due to attraction by successors and repulsion by predecessors, similar to the concentric field introduced by Sugiyama and Misue (1995).

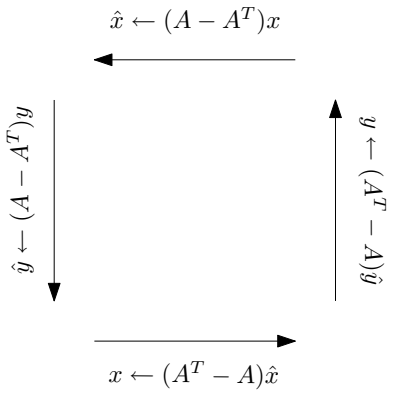


Figure 6.6: The rotation effect of multiplication with matrices $(A - A^T), (A^T - A)$. The arrows symbolize the movement due to attraction by successors and repulsion by predecessors.

6.3 Application to Tournaments

A special class of directed graphs is called *tournaments* (Harary and Moser, 1966; Reid and Beineke, 1978). A tournament $G = (V, E)$ on n nodes is an orientation of the complete undirected graph on n nodes. Tournaments are a model for round-robin competitions in which everybody competes with everybody else, and every competition $\{v, w\}$ has a winner v and a loser w, say, represented by the orientation of the edge $(v, w) \in E$.

Here we use a slightly less strict variant of tournaments, in which the underlying undirected graph is not required to be complete, because there are situations in which no winner can be determined. The method of clockwise drawing is applied to results of international football leagues in England, Germany, Italy, and Spain, in the seasons ending in 2006, 2007, and 2008. In every season, between every possible pair of teams two matches are carried out, each team being the home team once. The tournament graph contains an edge $(v, w) \in E$ when v dominates w, i.e., v has won more matches against w than v against w; ties are not considered.

Figure 6.7 shows drawings of all 12 tournaments, as given by the positions in the bimension of the largest eigenvalue. A cyclic structure is displayed in some of the configurations, such as England 2006/2007, Germany 2006/2007, and Spain 2006/2007, 2007/2008, which leads to the conjecture that these seasons were quite balanced, with no clear dominator. In these tournaments, some otherwise weak teams, which are dominated by most others, won against otherwise strong teams. For example, in the 2006/2007 season of the English Premier League, West Ham United (node on the lower left) closed the season on rank 15 of 20 teams, but dominated the champions Manchester United and fourth-ranked Arsenal FC.

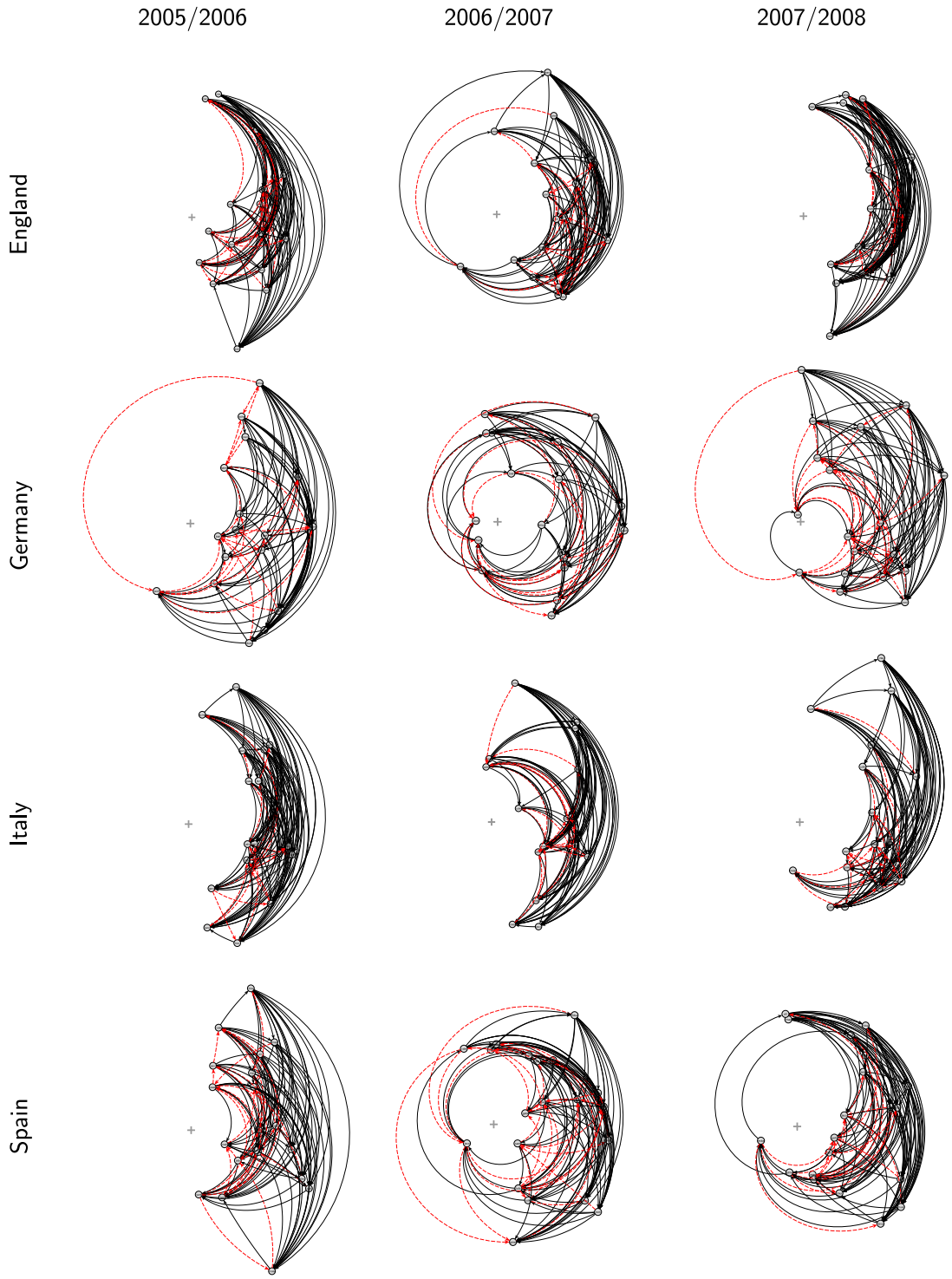


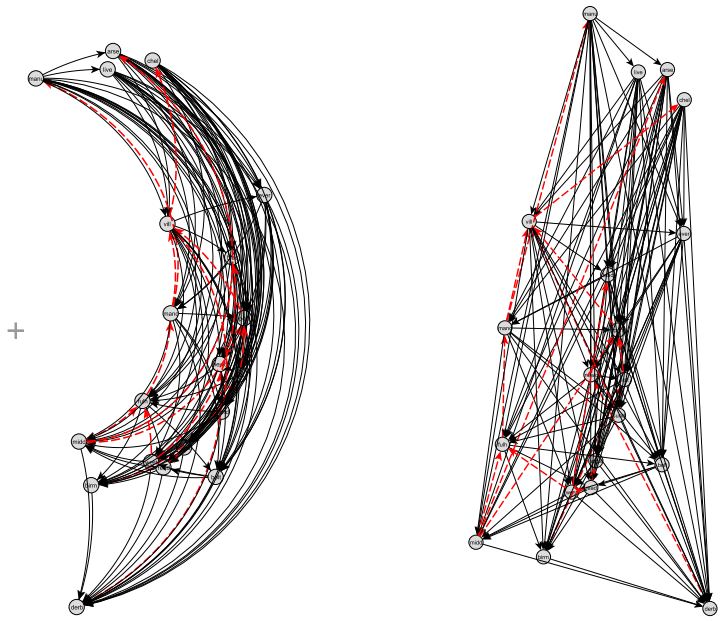
Figure 6.7: Clockwise drawings (bimension of the largest eigenvalue) of the tournament graphs in European football leagues in three consecutive seasons. The red dashed edges are counterclockwise.

Algorithm 10: Clockwise drawing of a directed graph

Input: directed graph $G = (V, E)$, accuracy $\epsilon > 0$
Output: positions x_v, y_v for all $v \in V$, eigenvalue ϕ
$x \leftarrow$ random, $y \leftarrow$ random
while *direction of* x *and* y *changes by more than* ϵ **do**
  $x \leftarrow x/\|x\|, y \leftarrow y/\|y\|$ // normalize
  $y \leftarrow y - \langle x, y \rangle \cdot x$ // orthogonalize
  foreach $v \in V$ **do**
    $\hat{x}_v \leftarrow \sum_{u \in N^-(v)} x_u - \sum_{u \in N^+(v)} x_u$ // $\hat{x} \leftarrow (A - A^T) \cdot x$
    $\hat{y}_v \leftarrow \sum_{u \in N^-(v)} y_u - \sum_{u \in N^+(v)} y_u$ // $\hat{y} \leftarrow (A - A^T) \cdot y$
  foreach $v \in V$ **do**
    $x_v \leftarrow \sum_{u \in N^+(v)} \hat{x}_u - \sum_{u \in N^-(v)} \hat{x}_u$ // $x \leftarrow (A^T - A) \cdot \hat{x}$
    $y_v \leftarrow \sum_{u \in N^+(v)} \hat{y}_u - \sum_{u \in N^-(v)} \hat{y}_u$ // $y \leftarrow (A^T - A) \cdot \hat{y}$
  $\phi \leftarrow \sqrt{\|x\|}$ // estimate for eigenvalue
$x \leftarrow x/\phi^{3/2}, y \leftarrow y/\phi^{3/2}$ // scale eigenvectors to length $\sqrt{\phi}$

In contrast, it is interesting to observe that the drawings of some other tournaments appear to be rather non-cyclic, especially England 2005/2006 and 2007/2008, all three seasons in Italy, and Spain 2005/2006. Since all nodes are on the same side of a line through the origin, the signed triangle areas do not allow for cyclic triples in this bimension. Thus, most of the dominance structure in the skew-symmetric adjacency matrix is intrinsically rather non-cyclic, which suggests that the classical hierarchical approach is actually more appropriate than the cyclic one. In the context of football matches, there is a high tendency for strong teams to consistently dominate weaker teams and weak teams to be consistently dominated by stronger teams, with no or only few exceptions.

In fact, a polar transformation easily transforms the half-circular arrangement into the classical hierarchical style; a natural ranking is given by the total order of angles of all nodes with respect to the origin, as given by their angular representation. The transformed coordinates are given by the transformation (6.15). α_v represents the transformed clockwise rank of v, and ρ_v the amount of skew-symmetry of v with all other nodes. An example of such a half-circular configuration with x-y coordinates and its polar transform with ρ-α coordinates is given in Figure 6.8.



(a) half-circular **(b)** hierarchical

Figure 6.8: Football tournament graph in England 2007/2008 (bimension of the largest eigenvalue). The graph exhibits a substantially hierarchical structure, which justifies the transformation from a polar domain (a) into a Cartesian domain (b). Note that this induces a correspondence of clockwise edges to downward edges.

6.4 Conclusion

Skew-symmetric scaling can be used for the layout of directed graphs in a cyclic fashion. Due to its algebraic nature, it provides direct and unique solutions. The drawing area is oriented either clockwise or counterclockwise around a distinguished center point; if necessary, the sense of rotation is inverted by reflecting one axis. The algorithm is easy to implement because it requires only basic array operations and no sophisticated data structures. Since no cycle removal or level assignment is required, some of the computationally hard problems are avoided.

When discrete levels or radial level assignments are required, they may be obtained from the continuous coordinates by a quantization scheme. The clockwise configurations can be combined with the force-directed methods in (Bachmaier et al., 2009; Sugiyama and Misue, 1995).

Extending the layout method to non-uniform edge lengths is straightforward. For strongly connected graphs the adjacency matrix may be replaced by the matrix of shortest-path distances, since all graph-theoretical distances are finite

and the corresponding skew-symmetric distance matrix is well-defined; the skew-symmetry between two nodes then indicates by how much faster or slower a node can be reached than vice versa.

Beyond the application to graph drawing, we expect that the presented method is also useful as a heuristic for to more general $\mathcal{NP}$-hard arrangement problems (Ganapathy and Lodha, 2004; Liberatore, 2004), and serves as a good starting solution, which is subjected to subsequent iterative improvements.

Part II

Applications

Chapter 7

Visual Analysis of Software Dependencies

From the developer perspective, a large software project typically comprises an "inside" region of source code created and maintained within a team of varying size, and an "outside" region of referenced and referencing modules, components, or libraries. Developers usually have a good local mental map of dependencies and references, such as compile-time dependencies, method calls, variable references, around their own source code. However, it is more difficult for them to keep track of the more general flow of dependencies coming from and going to the outside code world, especially in terms of transitivity, frequency, and importance.

Visualization can provide software architects and engineers insight into both dependencies and grouping among components in large software projects (Storey et al., 1996, 1997). A typical and frequent scenario in which such insight is needed arises when a software engineer joins a new team and needs to understand the structure of a current project (DeLine et al., 2005), or in reverse engineering or redesign of an entire software architecture (Müller et al., 1993).

We introduce a layout method which uses separate axes for the visualization of separate, yet not completely independent, aspects of a software system, which is modeled by dependency graphs:

- *Horizontal* positions reflect similarity and closeness of components. This accounts for *undirected* relations in the software system and conveys proximity, structural clusters, and symmetries.

- *Vertical* positions reflect the importance of components by a ranking corresponding to *directed* relations. It conveys overall trends and directions of dependency flows in the system.

In addition the method takes into account grouping information, which is either given by user input or derived from the graph meta-data. Parameters control the impact of this additional grouping information on the final layout for all these axes.

Rather than using a general method generating a layout for the a-posteriori interpretation and detection of the grouping and the component importance, our method incorporates an automatic a-priori analysis of these aspects, or user-input preferences, into the visualization.

Dependency graphs that occur in practice can be fairly large, in some cases containing tens of thousands of components. In such situations both the time and the space complexity are crucial. The method is based on solving systems of linear equations and scales to very large dependency networks with thousands of nodes. It yields unique and reproducible layouts and is easy to implement.

7.1 Dependency Graphs

Dependency graphs as formal model in software engineering have been introduced in (Podgurski and Clarke, 1990). They have been used for analyzing the structure of software systems (Orso et al., 2004), code optimization (Ferrante et al., 1987), testability (Jungmayr, 2002), and identification or prediction of instabilities and failures (Bevan and Whitehead, 2003; Nagappan and Ball, 2007; Schröter et al., 2006).

Recently, methods from the field of network analysis have been shown to be more useful for predicting defects in software systems than traditional complexity metrics (Zimmermann and Nagappan, 2008). The PageRank measure is applied to component dependencies and frequency of reuse in (Bay and Pauls, 2005).

There is a plethora of approaches for the layout of directed graphs (Bastert and Matuszewski, 2001). Perhaps the most widely-known is the Sugiyama framework (Sugiyama et al., 1981). Most methods focus on aesthetic criteria like edge crossing minimization and a small number of edges pointing downwards. While such approaches work well for small and medium graphs, they tend to create cluttered and unreadable layouts for larger graphs. Even more importantly, the underlying combinatorial problems often involve costly computations and thus inhibit interactive visualization. Other methods try to find good positions subject to given constraints derived from an a-priori analysis, e.g. separation and grouping (Dwyer et al., 2005, 2006b; Koren and Harel, 2005) or centrality (Brandes et al., 2003). The method presented in the following is largely inspired by the work on the visualization of search engine query results (Brandes and Cornelsen, 2003), of citation status in bibliographic networks (Brandes and Willhalm, 2002), and of directed graphs in general (Carmel et al., 2004).

The hierarchy information and the results of previous analysis steps are inegrated in visualizations, e.g., in the reverse engineering tool Rigi (Müller et al., 1993), as visual aids for manipulation and exploration of software systems (Storey and Müller, 1995), and for queries to software systems (Consens et al., 1992). Lanza and Ducasse (2003) present visual support for reverse engineering. The correlation between different aspects of hierarchies in a software system is visualized by Sawant and Bali (2007). In recent years, the task of displaying adjacencies in hierarchical data, which frequently arises in software visualization, has attracted more and more interest (Holten, 2006; Neumann et al., 2005).

7.2 Layout Algorithm

We model a software system as a directed graph $G = (V, E)$ of *components* or *nodes* $V = \{1, \ldots, n\}$, where $E \subseteq V \times V$, $m = |E|$, is a set of *dependencies* or *edges*. The *distance* d_{ij} between two components $i, j \in V$ is the length of a shortest undirected path of dependencies connecting i and j, where all edge lengths are one. We assume that G is weakly connected and thus that all d_{ij} are finite; otherwise we treat the connected components individually.

In the following, we will use A to denote the adjacency matrix of G and $D_- = \mathrm{diag}(\deg^-(1), \ldots, \deg^-(n)), D_+ = \mathrm{diag}(\deg^+(1), \ldots, \deg^+(n))$ to denote the diagonal matrices of in-degrees and out-degrees (the numbers of incoming and outgoing edges), respectively. Coordinates are written as n-dimensional column vectors $x = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$, where $x_i \in \mathbb{R}$ is the entry corresponding to component $i \in V$. By $\| \cdot \|$ we denote the Euclidean norm.

In the Subsections 7.2.1 and 7.2.2 we explain how the method works without the information about groups. Subsection 7.2.3 demonstrates how this grouping information can be integrated in the layout.

7.2.1 Horizontal Coordinates

Along the horizontal axis, similar objects should be placed close to each other and dissimilar objects should be separated. In this context, components are similar if they are connected by a short path of dependencies without regarding their directions. To this end, we use the one-dimensional solution of classical scaling, whose overall goal it is to approximate the dissimilarity of components by the Euclidean distance between them in the drawing.

For horizontal coordinates, which serve to reflect proximity in the undirected graph, a linear model is used, and components depending on the same components should be assigned similar positions. The desired configuration is written

as a vector $x = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$. For every pair i, j of components the target distance d_{ij} on a line is set to be the length of an shortest undirected path between i and j, using unit edge lengths. We try to find x such that

$$|x_i - x_j| \approx d_{ij} \tag{7.1}$$

is attained as well possible for all pairs i, j of components.

Let $\lambda_1 \geq \cdots \geq \lambda_n$ and $u_1, \ldots, u_n$ ($\|u_i\| = 1$ for all $i \in \{1, \ldots, n\}$) denote the sequences of eigenvalues and corresponding unit-length eigenvectors of $B = (b_{ij})$, having entries

$$b_{ij} = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{n}\sum_{r=1}^{n} d_{ri}^2 - \frac{1}{n}\sum_{s=1}^{n} d_{js}^2 + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} d_{rs}^2\right)$$

as in (3.5). The vector $x = \sqrt{\lambda_1}u_1$ is a global and, if $\lambda_1 > \lambda_2$, the unique solution of the optimization problem

$$\begin{aligned} &\sum_{i,j}(b_{ij} - x_i \cdot x_j)^2 \to \min \\ &\text{subject to } x \in \mathbb{R}^n, \quad \sum_{i=1}^{n} x_i = 0. \end{aligned} \tag{7.2}$$

Thus, the x-coordinates, which are assumed to be centered at zero, are a least-squares solution for fitting products $x_i \cdot x_j$, to the pseudo-products b_{ij} derived from the shortest-path distances d_{ij}.

Further dimensions for horizontal placement are obtained by subsequent eigenvectors of the matrices B as follows. Another dimension $y \in \mathbb{R}^n$ is found by orthogonalizing the current iterate $y^{[t+1]}$ against $x^{[t+1]}$ by

$$y^{[t+1]} \leftarrow y^{[t+1]} - \frac{x^{[t+1]T} y^{[t+1]}}{x^{[t+1]T} x^{[t+1]}} \cdot x^{[t+1]}. \tag{7.3}$$

as in Algorithm 2.

7.2.2 Vertical Coordinates

For vertical placement, we assess the importance of all components in the system. A component is regarded important if many other important components depend on it, and unimportant if no or only unimportant components depend on it. This feedback-based definition of importance is the basis for the well-known *PageRank* measure (Page et al., 1998) originally used for ranking search engine query results.

Given an a-priori importance distribution $\hat{p} \in \mathbb{R}^n$, whose entries sum to one, the PageRank is defined as the unique solution of

$$p = \omega \cdot M^T p + (1 - \omega) \cdot \hat{p}, \tag{7.4}$$

where $M = D_-^{-1} A$ is the adjacency matrix of G normalized so that all rows sum to one and $\omega \in [0, 1]$ is an escape parameter which is commonly set to 0.85. Unless stated otherwise, we use uniform a-priori probabilities $\hat{p} = [\frac{1}{n}, \ldots, \frac{1}{n}]^T$.

PageRank can be interpreted as the stationary probability distribution of a random walk along the edges of a directed graph: With probability ω, the random walker follows one of the outgoing links (with uniform probability); with probability $1 - \omega$, the random walker escapes to a random site, say i, with probability $\hat{p}_i$.

Since we convert PageRank into a geometric rank, an intuitive interpretation is that all components start on the same level, and dependent components "push up" other components. The higher the dependent component is, the more the other component benefits from being further pushed up.

PageRank is computed by initializing $p^{[0]} = \hat{p}$ and repeatedly setting

$$p^{[t+1]} \leftarrow \omega \cdot M^T p^{[t]} + (1 - \omega) \cdot \hat{p} \tag{7.5}$$

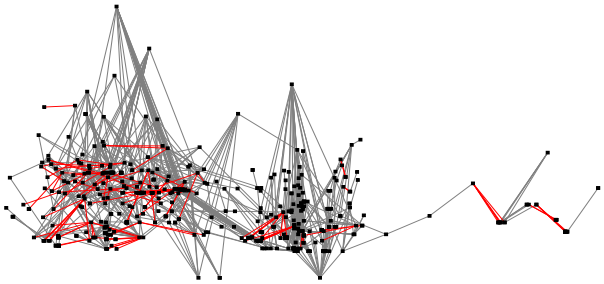
for growing $t \in \mathbb{N}$ until convergence. In practice, a small number of iterations (around 50) is sufficient. After convergence, we scale and translate p into the interval $[0, 1]$. Instead of carrying out the matrix multiplication $M^T p^{[t]}$, we exploit the sparsity of the graph and traverse each edge in the graph exactly once per iteration. The running time is in $\mathcal{O}(n + m)$, assuming a constant number of iterations.

A dual ranking is given by *reverse PageRank*, which is also called *TrustRank* (Gyöngyi et al., 2004). It is defined as the PageRank in the same graph with reversed edges. In (7.4) the matrix $M = D_-^{-1} A$ is replaced by the outdegree-normalized transpose adjacency matrix $D_+^{-1} A^T$. Intuitively, in the iterative computation again all components start at the same level. In contrast to PageRank, components are pushed up by unimportant components that depend on it.

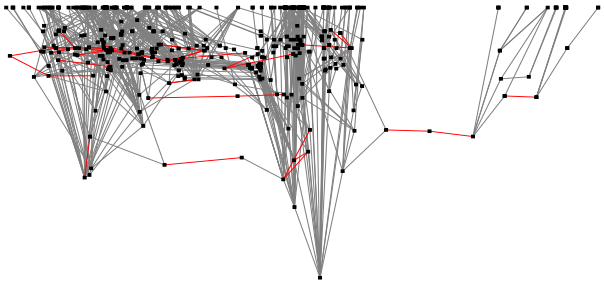
Figure 7.1 shows three drawings of a sample graph from the AT&T graph collection[1] with different vertical positions generated by the basic version of our method. In this and all other figures we have replaced the original ranking coordinates p_i by a monotonic transform $p_i^{1/3}$, which preserves the ranking of nodes but lessens the resolution problem of clustering nodes with a score close to zero. See also Dwyer et al. (2006a).

In Figure 7.1(a) most edges point upwards. To have the same effect for the reverse PageRank, the ranking is inverted by negating all values in Figure 7.1(b).

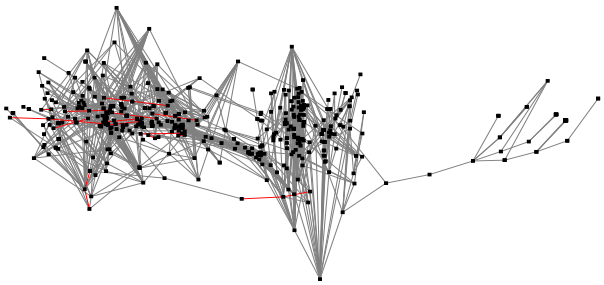
[1] `http://www.graphdrawing.org/data/`



(a) PageRank



(b) negative TrustRank



(c) average of both

Figure 7.1: Graph dg_3692 (559 nodes, 1 035 edges) from the AT&T digraph collection laid out with different vertical coordinates and identical horizontal coordinates using $\omega = 0.85$. Edges pointing downward are drawn in red.

Using the average of both variants has the effect of pushing very important nodes up and very unimportant nodes down, as depicted in Figure 7.1(c).

7.2.3 Adding Group Information

Typically, components in software systems are organized in packages, file systems, or namespaces. The basic approach, as described in the previous subsections, is now enhanced by modifying the computation of both horizontal and vertical positions according to this grouping information.

In our setting we use a function $g\colon \{1,\dots,n\} \to \{1,\dots,k\}$ that partitions the set of n components into k non-empty *groups*. By G_g we denote the directed *group graph* or *quotient graph* with the node set $\{1,\dots,k\}$ and directed edges obtained by collapsing the directed edges between groups, so the set of edges of G_g can be written as $\{(g(i),g(j)) : (i,j) \in E, g(i) \neq g(j)\}$.

In all axes, the final layout now becomes a compromise between the group level and the component level, the contributions of which are weighted by parameter $0 \leq \omega \leq 1$ ($\omega = 0$: no group-level information used, $\omega = 1$: only group-level information used), which corresponds to the escape parameter in the PageRank definition (7.4).

For the layout of our examples we used the same parameter for all dimensions, but it is possible to use different weights for each dimension. Furthermore, our two-level hierarchy approach can be extended to a multilevel layout for more complex hierarchies by applying it recursively.

Grouping for horizontal coordinates We use two competing objectives to reflect similarity: First, components should be close to their dependent components. Second, they should be placed close to other components in their group. The parameter ω weights how dominant the second objective should be in the linear combination.

Horizontal coordinates are obtained by modifying the input distances in (7.1) to *interpolated distances*

$$\hat{d}_{ij} = \omega \cdot d_{g(i)g(j)} + (1-\omega) \cdot d_{ij}, \tag{7.6}$$

a linear combination of group- and component-level distances. Here, $d_{g(i)g(j)}$ is the shortest-path distance between $g(i)$ and $g(j)$ in the underlying undirected graph of G_g.

Grouping for vertical coordinates In the first step, the group-level PageRank $p^g \in \mathbb{R}^k$ is computed on G_g using a uniform bias vector in (7.4). This is converted to a *component-level* PageRank distribution (all entries sum to one) for

individual components through

$$\hat{p}_i = \frac{p^g_{g(i)}}{\sum_{i=1}^{n} p^g_{g(i)}}, \tag{7.7}$$

so that each node i receives an a-priori bonus prominence inherited from its group $g(i)$.

In the second step, $\hat{p}$ is plugged into (7.4), where escape parameter ω can now be interpreted as the impact the group-level prominence has for generating the component-level layout. Note that for $\omega = 0$ the computation might break down because all importance is finally accumulated at a small set of sink nodes. This can be corrected by adding a small uniform escape probability.

Other applications of a-priori information in the PageRank computation can be found in (Haveliwala, 2003; Jeh and Widom, 2003; Kamvar et al., 2003; Padmanabhan et al., 2005). Langville and Meyer (2005) give a good survey of PageRank, related methods, and their computation.

7.3 Applications

To illustrate the method, we apply it to two different real-world dependency graphs. In the first example the grouping is automatically derived from metadata. In the second example components are manually tagged with group labels so as to create only few cross-group dependencies.

7.3.1 Java Standard Package

The layout of this graph was one of the challenges in the Graph Drawing Contest at the annual Graph Drawing Symposium in 2006. Unfortunately, there were no submissions in this category to which we could compare our results; the data set is publicly available (Duncan et al., 2007).

The graph has 1538 nodes representing Java classes and 8032 edges for compile-time dependencies. It contains all classes of the standard `java.*` package Java JDK 1.4.2 having some dependencies. Class i depends on class j if the compiler needs a compiled version of j to compile i. Note that this graph contains cycles in which all participating classes have to be compiled together. The grouping information is derived from the first level of the package namespace hierarchy, so there are groups `java.applet`, `java.awt`, `java.beans`, `java.io`, `java.lang`, etc.

Figure 7.2 and 7.3 show drawings of the graph with different values of grouping parameter ω. The area of the square representing a node is proportional to

the number of the incoming edges. The group labels are placed in the barycenter of the corresponding components. To reduce the clutter, individual component labels are only used for components with at least ten incoming edges. Dependencies within the same group are displayed by darker edges.

In Figure 7.2(a) grouping parameter ω is set to 0, indicating that the grouping information is not used for this drawing. The most important classes are, not surprisingly, `java.lang.String` and `java.lang.Object`, since almost all Java classes reference to it. A bit more surprising is the third rank of class `java.lang.RuntimeException`, despite its lower indegree; our explanation is that only a few classes throw runtime exceptions, but that these dependent classes are themselves important for many other dependent classes.

Figure 7.2(b) and Figure 7.3(b) are largely influenced by the grouping information determined by parameter $\omega = 0.5$ and $\omega = 0.9$, respectively. The higher ω is, the more apparent the group structure becomes at the expense of individual dependencies. The groups `java.awt`, `java.rmi`, and `java.sql` are rather separated from other groups, with mostly outgoing dependencies. A closer look at `java.awt` shows that `java.awt.Component` is very important in its group. In the center of the drawings there is no clear separation of the groups `java.util`, `java.net`, `java.lang`, and `java.io`, which seem to play a central role in the entire architecture. This is mainly due the majority of their members being in one strongly connected component. As above, `java.lang.String` and `java.lang.Object` are most important. A particularly interesting observation, which becomes more apparent for larger ω, is that `java.util.regex.Matcher` seems to be an important helper class.

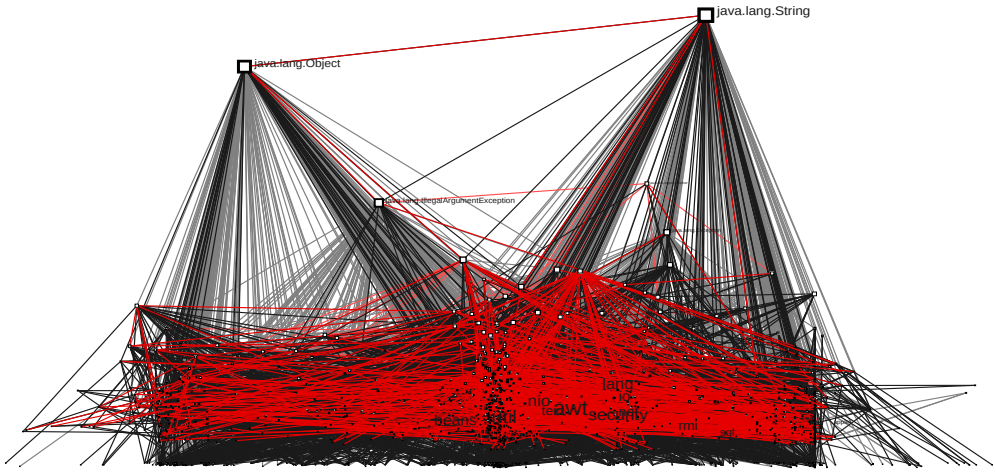
If ω is set to 1, the information about individual dependencies is discarded, and only dependencies between entire groups are displayed. As above, the groups `java.util`, `java.net`, `java.lang`, and `java.io` are so tightly coupled that they are assigned very similar positions at the very top of the whole group-level graph.

7.3.2 A Microsoft Software Project

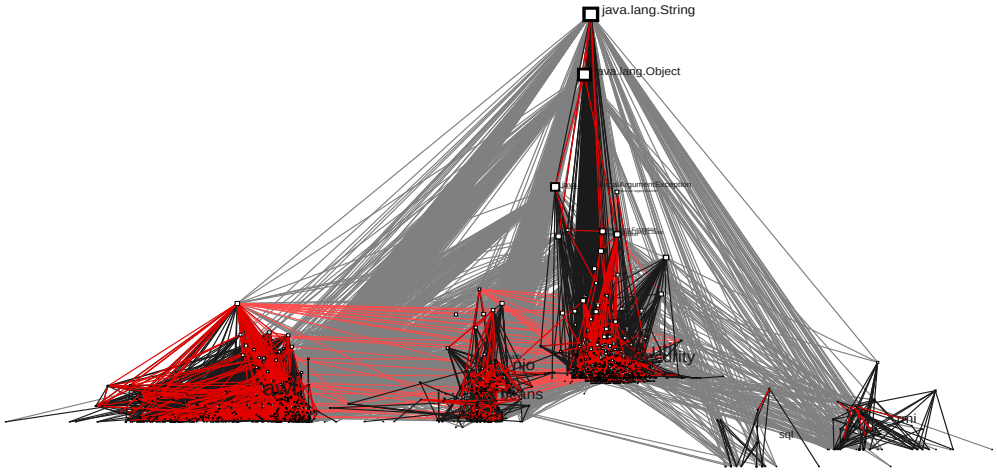
This data set describes the dependencies between all binaries of a large software project developed at Microsoft. It has 696 nodes for components and 1775 edges for dependencies. The components are organized in groups based on human knowledge. The objective of this grouping is to achieve a loose coupling of groups with few cross-group dependencies.

We created three-dimensional drawings of our graph for Figures 7.4 and 7.5. Each figure contains three aligned projections of a three-dimensional configuration.

The lower left parts of Figure 7.4 and 7.5 display the horizontal positions obtained from the first two eigenvectors of classical scaling. They can be seen

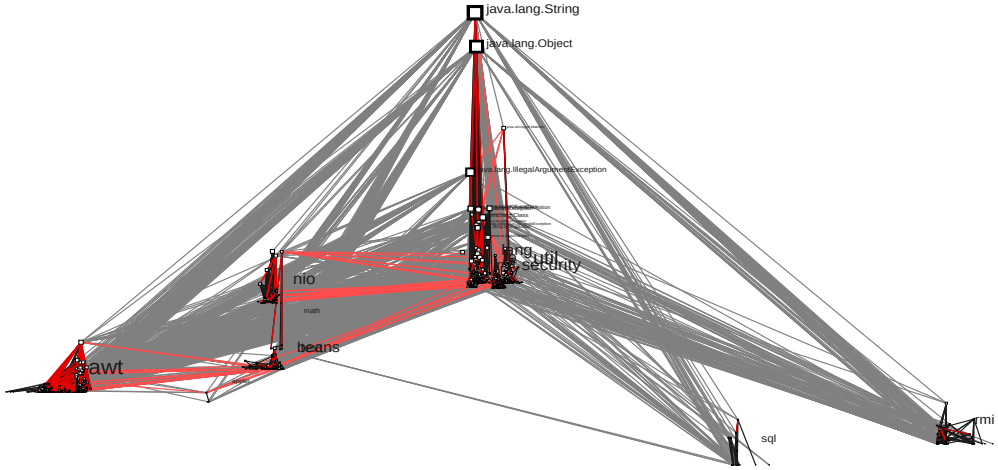


(a) $\omega = 0$: no group structure is visible

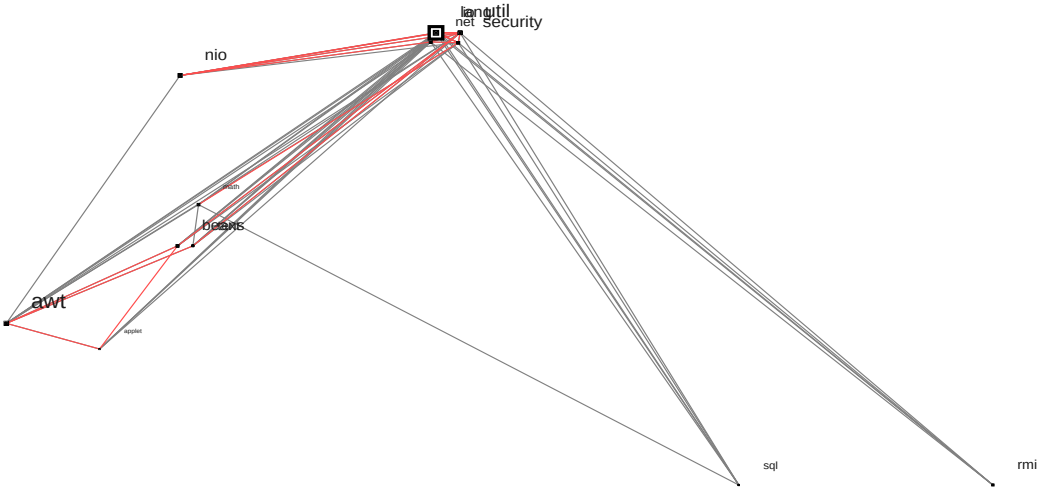


(b) $\omega = 0.5$: the group structure and major components within the groups are visible

Figure 7.2: Graph of Java compile-time dependencies (1538 nodes, 8032 edges) laid out using the basic method. Edges pointing downwards are visualized in red, darker edges are between components in the same group. The group labels are placed at the barycenter of the member components.



(a) $\omega = 0.9$. The group structure is visible, but the inner structure of groups is less visible



(b) $\omega = 1$, only the group structure is visible

Figure 7.3: Graph of Java compile-time dependencies (1538 nodes, 8032 edges) laid out using the basic method. Edges pointing downwards are visualized in red, darker edges are between components in the same group. The group labels are placed at the barycenter of the member components.

as a view from above and do not take dependency directions into account. If we compare them, we note that the modification of parameter ω can provide a visual cue for the quality of the grouping information. The vertical positions are derived from different rankings, as displayed in the upper and right parts.

In Figure 7.4 the influence of the grouping is very strong, with $\omega = 0.95$. Components in equal groups are placed close to each other, even if there is no dependency between them. PageRank is used for vertical positions to convey the overall importance of components. This highlights both the importance of entire groups and, within groups, the importance of individual components, as being mainly consumed.

The impact of grouping on the positions is somewhat weaker in Figure 7.5 with $\omega = 0.8$, while the dependencies across different groups have more influence in the layout. In effect, components are dragged towards similar components and possibly away from their group. Here, vertical positions are determined by using the negative values of the reverse PageRank. Unlike the vertical positions generated using PageRank in Figure 7.4, the reverse PageRank emphasizes components with many outgoing dependencies, the consumer components.

7.4 Conclusion

We have developed a method for laying out large directed graphs with additional grouping information. Such graphs frequently appear in software engineering as dependency graphs. Our approach supports the visual analysis of similarity and importance of components in large software systems. These aspects are visualized both at the group level and at the level of individual components.

Our main contribution is the automatic generation of meaningful positions according to a network analysis of the dependency graph. We are not aware of any other software visualization method with a similar a-priori analysis. We believe that our method is especially useful for graphs with more than 1000 edges, in which case the Sugiyama scheme and many other traditional methods tend to produce cluttered and unreadable layouts.

Layout approaches similar to our basic method, such as (Brandes and Cornelsen, 2003; Brandes and Willhalm, 2002), use energy minimizing properties of the graph Laplacian (Koren, 2005). They scale to large graphs, but their solutions tend to degenerate and the integration of grouping information is not straightforward; in fact, the Laplacian layout used there can be shown to be an instance of classical scaling; see Section 4.2. We prefer classical scaling to distance scaling because our experiments indicate that it does better in producing one-dimensional solutions, is more useful for reflecting structural clustering, and, above all, scales to much larger graphs.

The overall running time complexity of our method is in $\mathcal{O}(k(kn+m))$, which is essentially linear in the size of the graph if the number of pivots k is bounded by a constant. For instance, the complete layout computation of a dependency graph with about 10 000 nodes and 100 000 edges takes less than ten seconds on a standard notebook (Intel Pentium M with 1.6GHz, 512MB memory, using Microsoft Windows XP SP2 running Java JRE 1.6.0).

We make extensive use of the PageRank measure, but there is no limitation as to which importance measure to use for vertical positions. The analysis of software systems may be supported by other network analysis tools. Network flows, connectivity, automatic clustering, role models and other ranking methods than PageRank may be used to visualize further aspects of a software system, e.g. the identification of vulnerable components or instable subgroups.

Many software systems are not static but dynamically changing. Therefore, we think that it is promising to integrate the history of the software system into our method. For example, it could be interesting to see how components are promoted to the top of the layout because they become more important, or how the entire hierarchy changes over the course of software evolution.

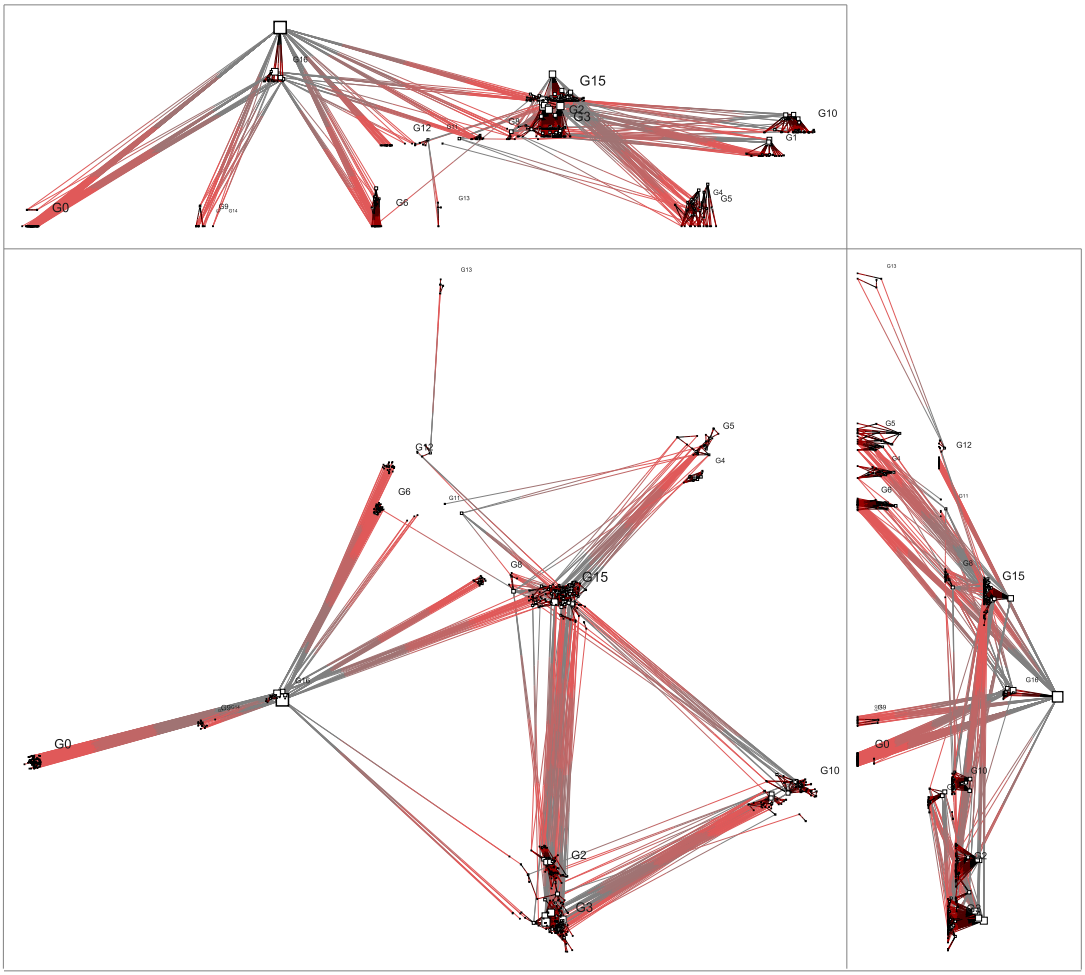


Figure 7.4: Three-dimensional layout of the dependencies in the Microsoft software project (696 nodes, 1775 edges). The use of PageRank ($\omega = 0.95$) for vertical coordinates emphasizes the importance of components by placing them on top of their dependent classes.

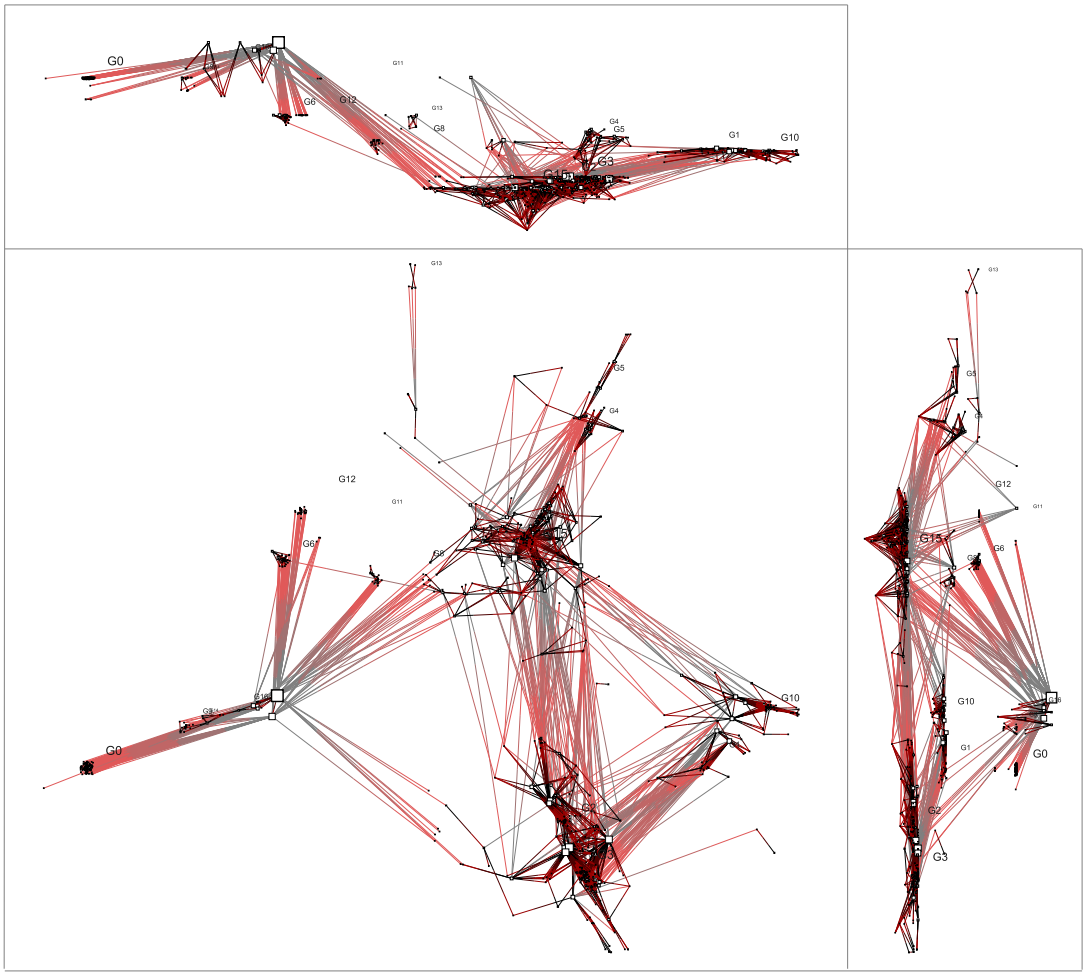


Figure 7.5: Three-dimensional layout of the dependencies in the Microsoft software project (696 nodes, 1775 edges). The use of the negative TrustRank ($\omega = 0.8$) for vertical coordinates emphasizes unimportant components, which are mainly "consumers", by placing them below the components they depend on.

Chapter 8

Radial Layout

In radial graph layouts the nodes are constrained to be located on a set of concentric rings; for some or all nodes in a graph a radius is given, which typically encodes results of a preceding analysis step. Those drawings date back to the 1940s and are called *ring* or *target sociograms* (Northway, 1940); there, four concentric ring-shaped regions represent quartiles and prescribe a region in which objects are allowed to be placed While the concrete meaning and the interpretation of a node being associated to a certain ring depends on the application at hand, the overall goals can be expressed in terms of the following two general criteria, which are possibly contradictory:

- *Representation of distances:* Nodes proximate in the graph should also be proximate in the drawing, and nodes distant in the graph should be distant in the drawing.
- *Radial constraints:* Nodes are associated with the radius of a circle centered in the origin, and are constrained to be placed on the circumference of this circle.

These criteria have guided the design of graph drawings in several applications. Radial layout is used for the exploration of large hierarchies (Wills, 1997); a hierarchy is laid out radially as a tree, followed by an incremental force-based placement. This approach was later modified by Yee et al. (2001) for dynamic real-time exploration of a filesharing network, where a user can interactively select a node, which is moved into the center, while the surroundings of that node are updated accordingly.

A different approach is taken by Bachmaier et al. (2005), where the definition of level planarity is extended to the case of radial discrete levels; the traditional Sugiyama framework is enhanced accordingly for the linear-time embedding of level-planar graphs.

In the case of continuous radii representing some kind of substance, unary constraints are imposed on the drawing for mapping centrality scores to visual centrality (Brandes et al., 2003). The layout is done by simulated annealing (Davidson and Harel, 1996), which allows for penalty costs, e.g. for edge crossings, and is very flexible but also computationally demanding, which prevents interactivity even for moderately sized graphs.

In the following, the two essential goals above are explicitly formulated as objective functions, which measure how far a layout is away from meeting the criteria, and which are sought to be minimized. While the first objective is captured by the traditional energy or stress measures, we will aim at meeting the second objective by introducing radial constraints into the stress-based layout model, by using a linear combination of the two objectives. The approach is a special case of the more general weakly constrained stress minimization layout discussed in Section 4.3.

Quite recently, extensions of the stress term have been used for drawing graphs with explicitly formulated aesthetic criteria, such as the uniform scattering of the nodes in a graph over a unit disk (Koren and Çivril, 2009), penalizing node overlaps (Gansner and Hu, 2009), or preserving a given topology (Dwyer et al., 2009).

All these approaches modify the *distances* themselves in one form or another, while the approach presented in this chapter is based on engineering the *weights* used in the stress minimization model. The weights are coefficients of error terms involved in the quality criteria to be minimized. If chosen carefully, they can be used to influence the configuration resulting from optimizing the stress function modified by them; see Figure 8.1 for an example. We are not aware of previous work which makes systematic usage of such a weighting scheme to take up a particular perspective on a graph in the layout.

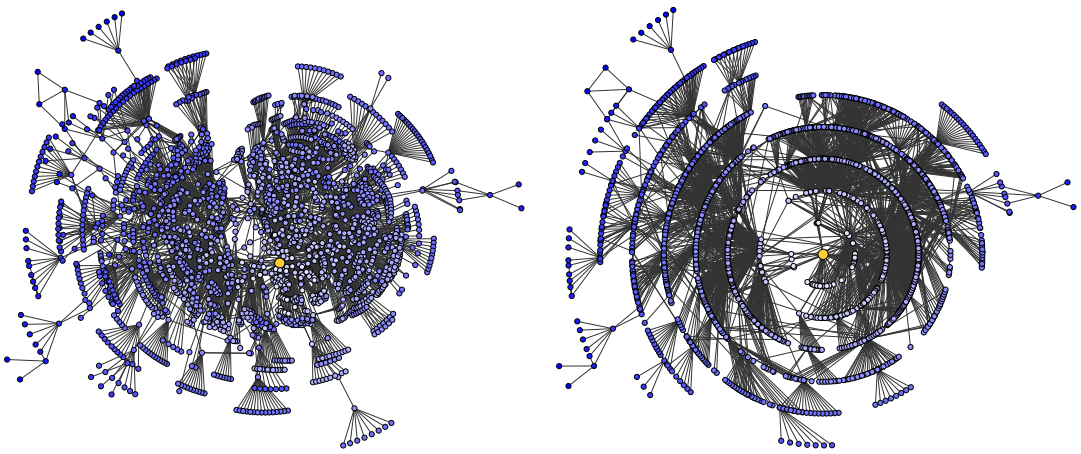
8.1 Stress, Weights, and Constraints

Node positions are denoted $p(v) = (x_v, y_v) \in \mathbb{R}^2$ and as vectors $x, y \in \mathbb{R}^n$. The Euclidean distance between the position of two nodes is denoted $\|p(u) - p(v)\| = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$.

8.1.1 Stress

Striving at the first of the two goal formulated above is simply formulated as the minimization of the weighted stress (discussed in Chapter 4)

$$\sigma(p) = \sum_{u,v} w_{uv}(d_{uv} - \|p(u) - p(v)\|)^2, \tag{8.1}$$



(a) unconstrained **(b)** with radial constraints

Figure 8.1: Social network (2075 nodes, 4769 edges) with two known clusters. The brightness of node colors is proportional to the shortest-path distances from a distinguished focal node, which also define the radii in the constrained layout, in which the clusters are still visible.

where the weight w_{uv} is set to d_{uv}^{-2} if $d_{uv} > 0$ and to 0 otherwise.

A reason for the favorable aesthetic properties of low-stress layouts is that no node is preferred over others because minimizing the objective function tries to balance the fit of desired distances. In most scenarios this is appropriate and tends to give the drawing a balanced appearance.

8.1.2 Weights for Constraints

In some cases, users want to put more emphasis on some nodes, while other nodes are regarded less important, by centering the view on a node and visualizing this node's neighborhood more prominently. This can be done by introducing suitable constraints to the configuration; when these constraints can be formulated as desired distances, choosing the weights in a suitable way allows for imposing them on the resulting layout.

What follows is a special case of the more general weakly constrained distance scaling presented in Section 4.3. While the range of possible applications is wide, our contribution will concentrate on the radial scenario. To avoid confusion, the objective function (8.1) will be termed *distance stress* and denoted $\sigma_W(p)$, where the subscript indicates that the stress is modified by a weight matrix $W = (w_{uv}) \in \mathbb{R}^{n \times n}$. This stress model is enhanced by a second weight matrix

$Z = (z_{uv})$ used in the *constraint stress*

$$\sigma_Z(p) = \sum_{u,v} z_{uv} \left(d_{uv} - \|p(u) - p(v)\|\right)^2, \tag{8.2}$$

whose minimization tries to fit the same distances and hence aims at representing the same information, but highlights different aspects.

8.1.3 Interpolated Weights

A straightforward approach to imposing the constraints expressed in a weight matrix is to directly minimize (8.2), but the resulting solutions tend to be trivial; for example, consider a linear layout in which $x_v = r_v, y_v = 0$ for all $v \in V$. Instead, it is more effective to combine distance and constraint stress into a joint majorization process, operating on a linear combination of the stress measures $\sigma_W(p)$ and $\sigma_Z(p)$.

Initially, the nodes are allowed to move freely without considering the constraints at all, by minimizing just $\sigma_W(p)$. Then, the constraints are granted more and more control over the appearance of the drawing by dynamically changing the coefficients in this combination, and the bias is shifted from one to the other criterion (Borg and Lingoes, 1980). The influences of two components are determined by the coefficients in the linear combination

$$\sigma_{(1-t)\cdot W + t\cdot Z}(p) = (1-t)\cdot\sigma_W(p) + t\cdot\sigma_Z(p), \tag{8.3}$$

and the update terms for the majorization process become

$$\hat{x}_v \leftarrow \frac{\sum_{u\in V\setminus\{u\}} \big((1-t)\cdot w_{uv} + t\cdot z_{uv}\big)\cdot\big(x_u + s_{uv}\cdot(x_v - x_u)\big)}{\sum_{u\in V\setminus\{v\}} \big((1-t)\cdot w_{uv} + t\cdot z_{uv}\big)}, \tag{8.4}$$

$$\hat{y}_v \leftarrow \frac{\sum_{u\in V\setminus\{v\}} \big((1-t)\cdot w_{uv} + t\cdot z_{uv}\big)\cdot\big(y_u + s_{uv}\cdot(y_v - y_u)\big)}{\sum_{u\in V\setminus\{v\}} \big((1-t)\cdot w_{uv} + t\cdot z_{uv}\big)}, \tag{8.5}$$

where

$$s_{uv} = \begin{cases} \frac{d_{uv}}{\|p(u)-p(v)\|} & \text{if } \|p(u)-p(v)\| > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8.6}$$

In the majorization, the bias is shifted from the distance component towards the radial component by gradually increasing t from 0 to 1.

8.2 Applications

8.2.1 Target Diagrams

A node is put in the *focus* by emphasizing the visual display of its vicinity, constraining all others to attain Euclidean distances corresponding to their graph-theoretical distances. That is, relative to the focused node, all structural distance-k neighborhoods are mapped to the geometric k-neighborhood.

The constraint weight matrix takes only pairs of nodes into account in which the focused node is involved, while reducing all other weights to zero; let f denote the focused node. D and W are defined as above, and the constraint weight matrix $Z = (z_{uv})$ has only zero entries except for the particular row and column containing the weights derived from the distances to f,

$$z_{uv} = \begin{cases} w_{uv} & \text{if } f \in \{u, v\}, \\ 0 & \text{otherwise,} \end{cases} \tag{8.7}$$

so that interpolating from W to Z gradually increases the focal node's relative impact on the configuration.

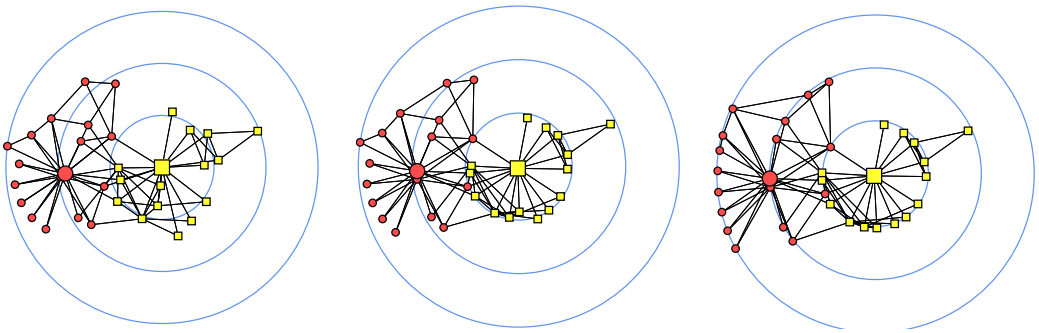
A famous social network was studied by Zachary (1977) and, subsequently, many others. It describes the friendship relations among 34 members in a karate club in a US university in the 1970s. Over the course of a two-year study, the network breaks apart into two clubs because of disagreements between the administrator and the instructor, who leaves the club and takes about half of the members with him.

Figure 8.2 shows how an initial layout is gradually modified to different radial layouts focused on the instructor (a) and the administrator (b). Eventually, every node is located on a ring with radius equal to its distance to the focused node. Since the original structural properties are also based in shortest paths, the radial and distance constraints are naturally compatible.

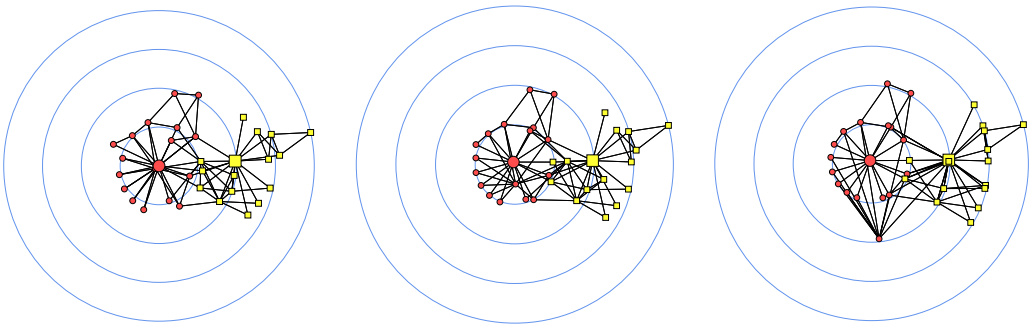
8.2.2 Centrality Drawings

When the radial constraints do not directly correspond to one of the columns in the distance matrix, we assume that nodes are numbered $v_1, \ldots, v_n$ and that the radii are given as additional input by a vector $r = [r_1, \ldots, r_n]^T \in \mathbb{R}^n$, with $r_i \geq 0$ for all $v_i \in V$. The radial constraints can be formulated in terms of distances to the center; since node v_i is located on a circle with radius r_i if its Euclidean distance to the center is equal to r_i, this is equivalent to

$$\|p(v_i)\| = r_i. \tag{8.8}$$



(a) focusing on the instructor



(b) focusing on the administrator

Figure 8.2: Radial layouts of Zachary's karate club network ($n = 34, m = 77$), by weight interpolation, for $t \in \{0, 0.9, 1\}$. Members leaving with the instructor are shown as yellow squares, members staying with the administrator as red circles.

The origin is treated as an additional dummy node o indexed with $n+1$, with artificial desired distances $d_{vo} = d_{ov} = r_v$. The stress majorization procedure is applied to a layout problem of $n + 1$ objects. Borg and Lingoes (1980) use such a dummy to enforce a circular configuration by using the same radius for all objects. The distance and weight matrices involved in (8.3) are

$$D = \begin{bmatrix} d_{v_1v_1} & \cdots & d_{v_1v_n} & r_1 \\ \vdots & \ddots & \vdots & \vdots \\ d_{v_nv_1} & \cdots & d_{v_nv_n} & r_n \\ r_1 & \cdots & r_n & 0 \end{bmatrix}, W = \begin{bmatrix} d^{-2}_{v_1v_1} & \cdots & d^{-2}_{v_1v_n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ d^{-2}_{v_nv_1} & \cdots & d^{-2}_{v_nv_n} & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix}, Z = \begin{bmatrix} 0 & \cdots & 0 & r_1^{-2} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & r_n^{-2} \\ r_1^{-2} & \cdots & r_n^{-2} & 0 \end{bmatrix}$$

Let c_v be a centrality measure on the nodes of graph G. The radius for every

node $v_i \in V$ is given by

$$r_i = \frac{\operatorname{diam}(G)}{2} \cdot \left(1 - \frac{c_{v_i} - \min\limits_{u \in V} c_u}{\max\limits_{u \in V} c_u - \min\limits_{u \in V} c_u + c(G)} \right). \tag{8.9}$$

Distances and the radial constraints are brought onto the same scale, e.g., by multiplying the radii with $\operatorname{diam}(G)/2$. The parameter $c(G)$ is a small offset which serves to prevent a set of more than one maximally central node from coinciding in the center (Brandes et al., 2003). Different regions are emphasized by alternative definitions of radii. Instead of the plain radii in (8.9), the central (peripheral) areas are enlarged by applying a concave (convex) function magnifying the regions of smaller (larger) centrality scores.

Examples of centrality drawings for Zachary's karate club network are shown in Figure 8.3. The left column contains drawings using the *betweenness centrality* (Anthonisse, 1971; Freeman, 1977)

$$c_v = \sum_{u \neq v \neq w \in V} \frac{\delta_{uw,v}}{\delta_{uw}} \tag{8.10}$$

where δ_{uw} denotes the number of shortest paths between u and w ($u, w \in V$,$u \neq w$), and $\delta_{uw,v}$ the same but restricted to paths that have v as a proper inner node ($v \in V \setminus \{u, w\}$); by convention, $\delta_{vv} = 1$. The right column shows the normalized *closeness centrality* (Sabidussi, 1966)

$$c_v = \frac{1}{\sum\limits_{u \in V} d_{uv}}. \tag{8.11}$$

Simplified pseudo-code, which is targeted at general radial constraints, is given in Algorithm 11. The quantities a_i are defined similar to (4.6) as

$$a_v = \begin{cases} \frac{1}{\|p(v)\|} & \text{if } \|p(v)\| > 0, \\ 0 & \text{otherwise} \end{cases} \tag{8.12}$$

For dynamic visualization scenarios, an inherently smooth transition between layouts with different foci is obtained by simply using the intermediate layouts given by the steps in the majorization process. In the transition from one focus to the other, it is advantageous to not directly interpolate between the two corresponding constraint weight matrices, but to take a detour via the original weight matrix having entries d_{uv}^{-2}, so as to re-introduce all the shortest-path distances to remove artifacts potentially introduced after focusing on the first node.

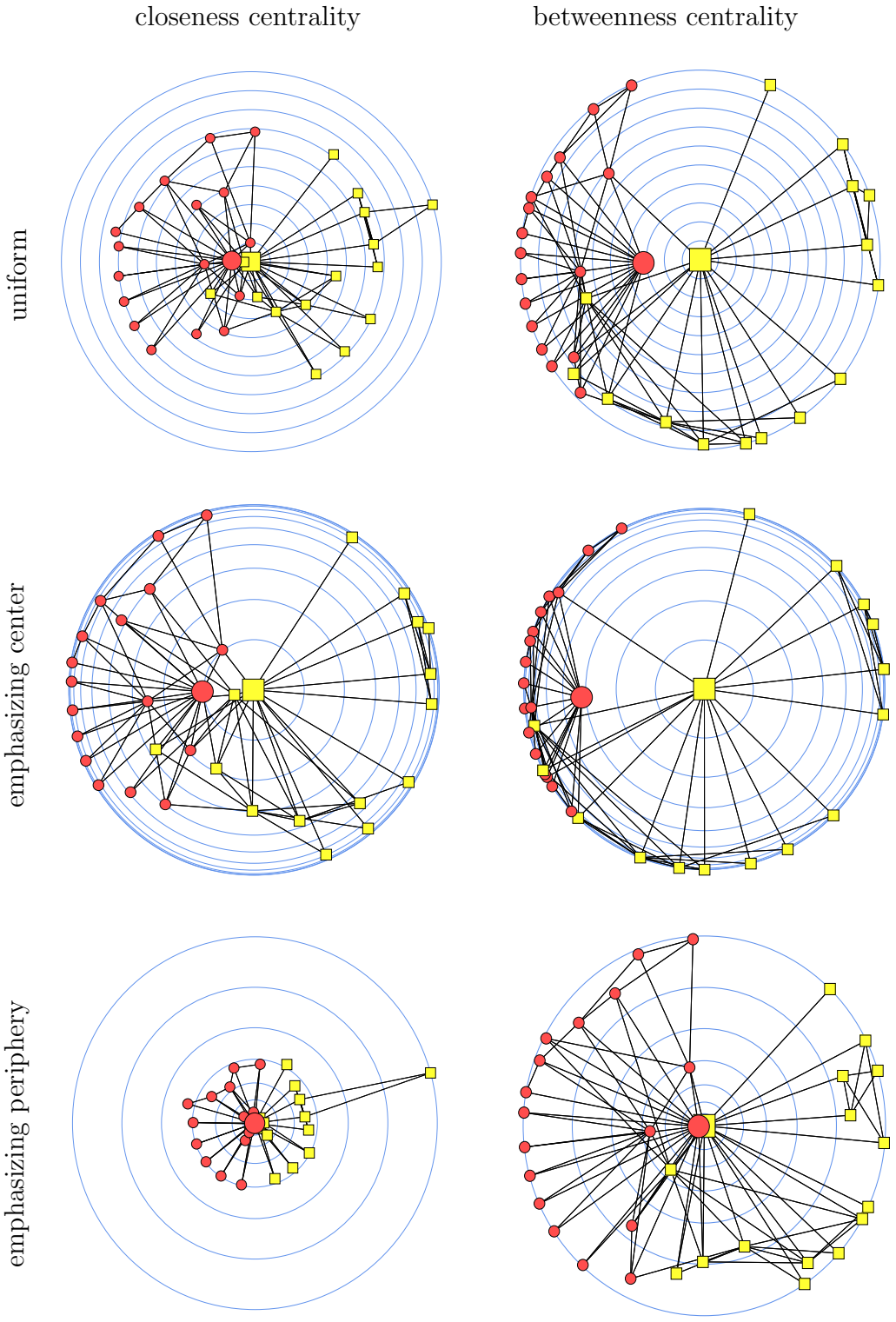


Figure 8.3: Centrality layouts of the karate club social network, using two centrality measures to define the radii of nodes. Nodes with high centrality scores are emphasized by $r'_i = 1-(1-r_i)^3$, the periphery of nodes with low scores by $r'_i = r_i^3$ and replacing r by r' ($0 \leq r_i \leq 1$ for all i).

Algorithm 11: Radial layout

Input: connected undirected graph $G = (V, E)$, radii $r_v \in \mathbb{R}_{>0}$ for all $v \in V$, number of iterations $k \in \mathbb{N}$

Output: coordinates $p(v)$ with $\|p(v)\| = r_v$ for all $v \in V$

$D \leftarrow$ matrix of shortest path distances d_{uv}
$W \leftarrow$ matrix of inverse squared distances d_{uv}^{-2}
$p \leftarrow$ layout with low $\sigma_W(p)$
for $t = 0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1$ **do**
  foreach $v \in V$ **do**

$$x_v \leftarrow \frac{\sum_{u \in V \setminus \{v\}} (1-t) \cdot w_{uv} \big(x_u + d_{uv} \cdot (x_v - x_u) \cdot b_{uv}\big) + t \cdot r_v^{-2} (r_v x_v a_v)}{(1-t) \sum_{u \in V \setminus \{v\}} w_{uv} + t \cdot r_v^{-2}}$$

$$y_v \leftarrow \frac{\sum_{u \in V \setminus \{v\}} (1-t) \cdot w_{uv} \big(y_u + d_{uv} \cdot (y_v - y_u) \cdot b_{uv}\big) + t \cdot r_v^{-2} (r_v y_v a_v)}{(1-t) \sum_{u \in V \setminus \{v\}} w_{uv} + t \cdot r_v^{-2}}$$

8.2.3 Travel Time Maps

When traveling with a public transportation system, schematic maps are essential for many users. Such maps depict lines, stations, zones, and connections to other traffic systems. Since the primary use for such a map is travel planning, usability and readability are more important criteria than the accurate representation of actual geographic positions. In graph drawing, this style is called *metro map layout* (Hong et al., 2004).

One of the most prominent schematic maps is Beck's famous London tube map; it has been and is still being reworked and improved and has inspired similar maps for systems of public transportation all over the world (Garland, 1994). While schematic maps are widely perceived as very useful, a potential drawback is that they tend to distort a user's perception of proximity, thus compromising the decisions made in the travel planning process, e.g., because stations are displayed as more proximate than they actually are.

If the starting and ending stations of a planned journey are known, the radial constraints can be used to highlight the time needed for traveling between them, by focusing only on one, as described above. In addition, shortest paths between the two stations can be highlighted by putting the focus on both of them at the same time.

Let the nodes in the focus be f_1 and f_2. Again, $D, W \in \mathbb{R}^{n \times n}$ are defined as the matrices of shortest-path distances and their inverse squares, respectively.

The weight matrix is $Z = (z_{ij}) \in \mathbb{R}^{n \times n}$ with

$$z_{uv} = \begin{cases} w_{uv} & \text{if } u \in \{f_1, f_2\} \text{ or } v \in \{f_1, f_2\} \\ 0 & \text{otherwise} \end{cases} \tag{8.13}$$

and contains a $(n-2) \times (n-2)$ submatrix with zero entries.

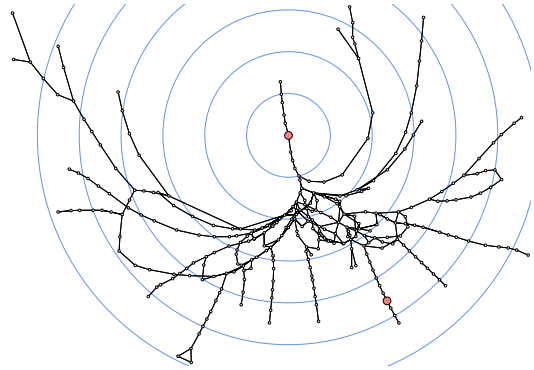
We use the connection graph of the London tube with estimated minimal travel times obtained from the Transport for London web site `www.tfl.gov.uk`, or derived from the geographic distance, where estimates are not available. For the sake of simplicity, walking times needed for changing lines were not considered; see also Carden (2005). Radial layouts are given in Figure 8.4, where stations are placed at a distance from the center proportional to their estimated minimum travel times from (a) Golders Green and (b) Greenwich independently and (c) from both stations at the same time.

8.3 Conclusion

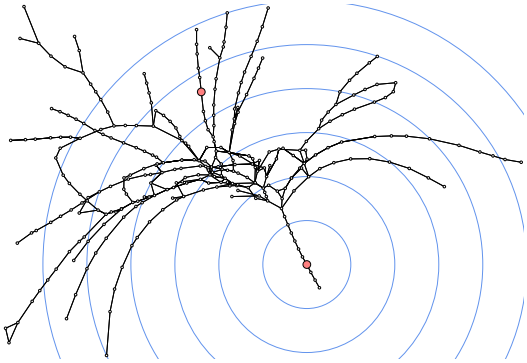
Radial constraints fit well into the framework of multidimensional scaling by stress majorization because the radii can be expressed in terms of desired Euclidean distances, which requires only minor modifications of available implementations of the stress minimization.

Motivated by the results obtained from the comparably simple radial constraints, and other experiments, we feel that they deserve more attention, because they allow the aesthetic goals of the visualization results to be explicitly formulated and quantified, and can be easily plugged into existing algorithms.

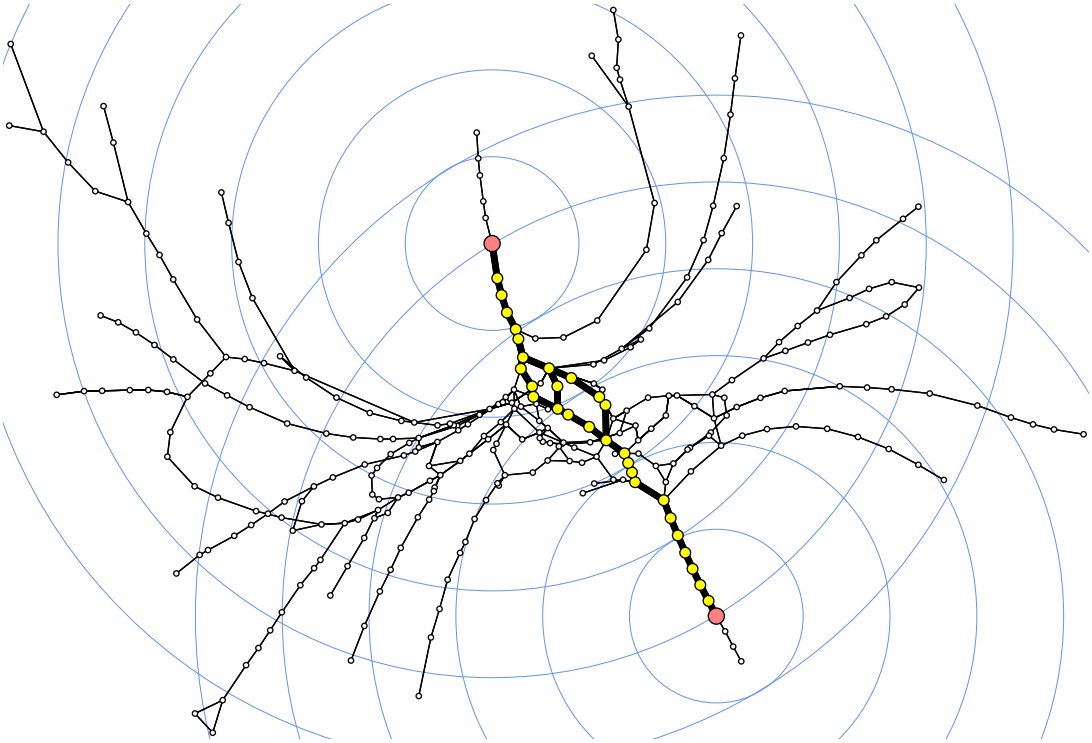
With the careful choice of a weighting scheme, the ideas presented above are easily carried over to layout tasks with more general constraints, such as the display of grouping, the computation of dynamic layouts, and the visualization of edge strength, certainty, or probability.



(a) travel time from Golders Green



(b) travel time from Greenwich



(c) travel time between Golders Green and Greenwich

Figure 8.4: Radial layouts of the London Tube graph using estimated travel times (307 nodes, 410 edges). The concentric circles indicate travel times in multiples of 10 minutes. The stations are constrained to be at distance equal to their minimum travel times.

Chapter 9

Layout of Bibliographic Networks

The applications in which data objects are related to or affiliated with other objects are countless. Networks are a frequent model for this kind of data, and their analysis often relies on good visualizations, which can be helpful for understanding structural properties and designing further analytic tasks. For example, visualization can aid in supporting or discarding hypotheses about clusters, patterns, linking behavior, importance analysis, and the underlying processes leading to the observed network.

Among the most popular analytic aspects of network data are proximity, similarity, and distance. Consequently, they are often the basis for network visualizations. While there is a plethora of such approaches, many of them are inherently limited to small or medium data sets and impractical for larger ones.

In this chapter we present an application of classical scaling (see Chapter 3) to the layout and visual analysis of large-scale multimode networks. The fundamental notion is proximity among items in the network to be laid out. This proximity, whose concrete definition depends on the particular network at hand, is represented by geometric proximity through positioning objects on a two-dimensional drawing area in a suitable way.

The method is applied to networks of bibliographic objects and associations (White and McCain, 1989) and their visualization (Brandes and Willhalm, 2002; Chen and Hsieh, 2007; Small, 1999). Nodes of various types, such as articles, authors, institutions, and journals, are linked by edges of different types, such as citations, authorship, and affiliation. The example data set is the result of a query to a large scale bibliography database. One of the main objectives of this chapter is to demonstrate that MDS is scalable even to very large graph drawing tasks.

9.1 Bibliographic Data Model

This section describes the underlying bibliographic data model for this type of network: The *objects*, which correspond to entities in bibliographic data; the basic *relationships* among these objects, as they are stored in bibliographic databases; and the bibliographic *operators* that transform relationships into other relationships dedicated to different analytic perspectives on the data.

Citations and references in articles and other research contributions are of increasing interest in the structural analysis of scientific literature. The number of times an article is cited and the origins of the citations serve as important proxies for its influence on subsequent work; they are often used for assessing the impact and the evolution of this article, its authors, its authors' institutions, the journal in which it appears, and even its entire scientific discipline.

In this chapter we will use simple lower case letters (m, n, p) for numbers, indexed lower-case letters (w_i) to denote objects, script letters $(\mathcal{C}, \mathcal{W}, \mathcal{A})$ for sets of objects, and upper-case letters (G, C, A) to denote matrices that represent relationships between objects.

9.1.1 Objects

Bibliographies are modeled by multimode networks in terms of sets of *objects* of different types, together with *relationships* between them. In this article the model and the bibliographic analysis encompasses sets of

- *authors* $\mathcal{A} = \{a_1, \ldots, a_m\}$,
- *works* $\mathcal{W} = \{w_1, \ldots, w_n\}$, and
- *journals* $\mathcal{J} = \{j_1, \ldots, j_p\}$.

This basic model may be extended by information about affiliations of authors to institutions, scientific disciplines, keywords attached to works, etc., but we will restrict the description to these three types.

9.1.2 Basic Relationships

The links connecting bibliographic objects, e.g. citations and affiliations, are conveniently modeled as matrices. A basic relationship is the form in which the data are stored in bibliographic databases and is typically available directly without much additional effort.

In our bibliographic analysis we distinguish three principal relationship types and thus network types. Basic relationships are modeled as unweighted graphs

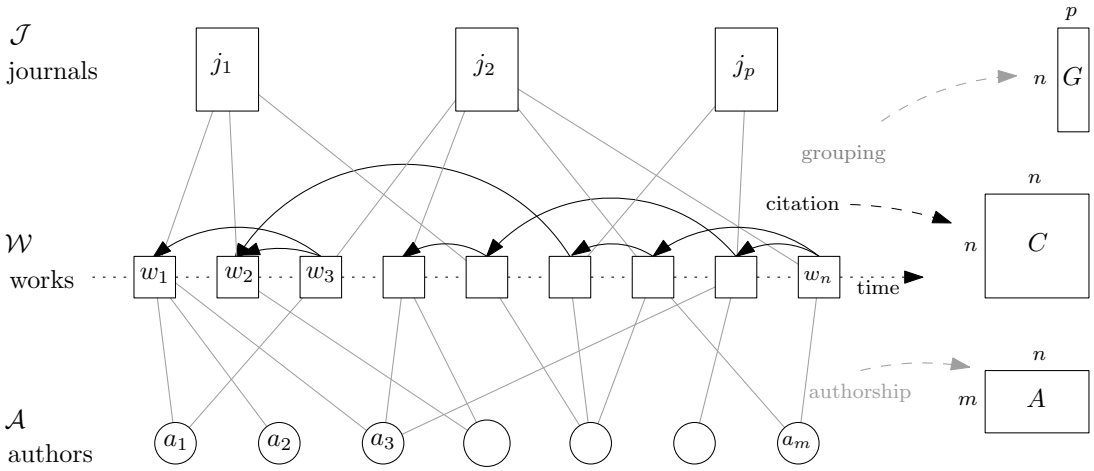


Figure 9.1: The basic data model for bibliographic networks. The matrices on the right represent basic relationships: A bipartite journals–works graph; a directed acyclic graph of citations among works; and a bipartite works–authors graph.

represented by adjacency matrices with entries 1 if the two corresponding objects are related, and 0 otherwise. Figure 9.1 depicts the objects and basic relationships in our bibliography model.

- *Citations* among works are stored in a square adjacency matrix $C \in \{0,1\}^{n \times n}$, representing a directed graph, with a directed edge from work w_i to work w_j if and only if w_i cites w_j. Citation graphs are usually inherently acyclic because cited works have to be published previously.

- *Authorship* is expressed in a rectangular two-mode adjacency matrix $A \in \{0,1\}^{m \times n}$. It represents a bipartite graph containing an edge from author $a \in \mathcal{A}$ to work $w \in \mathcal{W}$ if and only if a is an author of w.

- *Grouping* is information about journals in which works are published; institutions to which authors are affiliated; or scientific disciplines assigned to journals. Assuming that every work appears in exactly one journal, every row of matrix $G \in \{0,1\}^{n \times p}$ contains exactly one 1.

9.1.3 Bibliographic Operators

Bibliographic operators combine the previously introduced basic relationships and transform them into new relationships. In terms of matrices, the operators correspond to multiplication of the involved matrices.

In the following, we name some frequent bibliographic operators, which will later be used for visualization. Each operator reflects a particular analytic per-

spective on the basic relationships in the original bibliographic data. Note that the resulting matrix products are themselves matrices, which, in turn, represent graphs. The following is just a selection of the possible bibliographic operators; many others have been defined and used.

- *Bibliographic Coupling* between two works occurs if they have cited references in common (Kessler, 1963). The ij entry in $CC^T \in \mathbb{N}^{n \times n}$ is the number of common references of works w_i, w_j.
- *Co-Citation* is the dual to bibliographic coupling, measuring for two works w_i, w_j how many other works cite both of them, given in matrix $C^T C \in \mathbb{N}^{n \times n}$ (Small, 1973).
- *Collaboration* counts the number of common works of two authors. $A^T A \in \mathbb{N}^{m \times m}$ is a well-known operator, giving the number of times two authors have collaborated in a common work.
- *Projection* converts the citation information among works to the citation information among authors. The analysis is targeted at citations of authors (White and Griffith, 1981); the required information is obtained by the matrix product $ACA^T \in \mathbb{N}^{n \times n}$. The entry ij in this matrix is the number of times author i cites author j.

In an optional post-processing step, it might be useful to discard, e.g., the counts of co-authorship, and to replace all positive entries in the resulting matrix products simply by ones, or to normalize entries such that rows sum to one.

9.1.4 Bibliographic Proximity

Bibliographic relationships are the basis for calculating proximity. Traditionally, visualizations use similarity measures of binary attribute sets, such as the relative overlap of the number of citations (He and Hui, 2001; White and McCain, 1998). In the visualization setting to be described later, similarity measures have some shortcomings (Gower and Legendre, 1986). Their typical range from 0 to 1 often leads to degenerate visualizations, since many of them assess works with no common references as maximally dissimilar (similarity 0, dissimilarity 1). We replace the commonly used similarity of neighborhood sets by graph-theoretical distances in the derived bibliographic networks.

Here and later in Section 9.2 we focus on the bibliographic coupling operator, but the definitions are carried over in a straightforward way to the other operators defined above, e.g., to obtain a proximity based on *being cited* rather than *citing* the same or similar works. The motivation for using the bibliographic

coupling is that works with a large reference list overlap are regarded as similar and thus proximate.

Coupling graph Let $\mathcal{W}_i, \mathcal{W}_j$ denote the set of works cited by w_i and w_j, respectively. The bibliographic coupling is given by CC^T; the entry ij is the number of works cited by both w_i and w_j, and the edges in the corresponding *coupling graph* indicate that two adjacent works have at least one common reference.

A large number of common references should be reflected by higher proximity. Therefore, edges in the coupling graph have length inversely proportional to the strength of co-citation; the edge (w_i, w_j) has length proportional to the *absolute frequency* $1/|\mathcal{W}_i \cap \mathcal{W}_j|$, i.e., by regarding two works as more proximate if the absolute frequency of common citation is large.

Alternatively, edge lengths may be combined with the above-mentioned similarity measures, such as the *relative frequency* $|\mathcal{W}_i \cup \mathcal{W}_j|/|\mathcal{W}_i \cap \mathcal{W}_j|$, whose inverse is referred to as the *Jaccard index* (Jaccard, 1901).

Significant ties To make distances more robust against degeneration, e.g. due to standard works cited in almost all other works, we consider only *significant* co-citations by discarding co-citation frequencies below a predefined threshold. This makes the resulting distances more appropriate while reducing the set of edges, leading to more efficient distance computations in practice.

Although thresholding potentially results in disconnected graphs, which requires every connected component to be analyzed and visualized separately, our experience with real-world citation data sets is that the major connected component contains almost all works, while the small components contain only few works, which are less interesting in the analysis and the visualization.

Coupling distance Using only the plain number of common citations restricts the similarity to local relations. For example, consider the works w_i, w_j, w_k, where w_i has one half of its cited references in common with w_j and the other half in common with w_k; when w_j and w_k have no or only few references in common, they are considered maximally dissimilar, even though the relation between w_j and w_k might be particularly interesting, e.g. because w_i is a survey article which bridges different fields of research.

To be able to identify such bridges, the notion of similarity induced from the common citations is extended to a transitive, more global measure of *bibliographic proximity* of works w_i, w_j. Formally, it is defined as the graph-theoretical distance between w_i and w_j in the graph corresponding to the co-citation matrix CC^T, i.e., the length of a shortest path along bibliographic coupling edges that have to be traversed to get from w_i to w_j.

Note that it is often more computationally advantageous not to explicitly construct the inherently dense co-citation graph but to compute the distances in it by traversing the original sparse directed citation graph. Such cases require special care when implementing the required graph traversal procedures.

9.2 A Bibliography of Social Network Analysis

We illustrate our approach with an application to the visualization of citation data in Social Network Analysis literature and focus on bibliographic coupling and the similarity of citation behavior within a set of some thousands of items.

The data used in our application were obtained from the Web of Science, a commercial citation database run by Thomson Reuters.[1] The WWW interface allows for queries to several thousands of scientific journals, with the usual search criteria, such as author name, title, year, etc. From a set of query results, e.g. all works with a title containing a given string, the query interface allows for retrieval of the resulting sets of citing or cited references. Using these results, the bibliographic networks are created. Query results may be combined by union or intersection operators.

This data set, called *SN5*, was prepared by Vladimir Batagelj.[2] It contains the results of the query

```
(‘‘social network*’’ AND SO=(Social networks))
```

which gives all works in the data base with "social network*" in the title, together with all articles published in the *Social Networks* journal. In addition, this set of works is enriched by the most frequently cited works in the social network analysis literature and the works of 100 prominent researchers in the field of social network analysis, as far as not already contained in the results of the original query. The result is denoted as $\mathcal{W}$, the initial set of works.

9.2.1 Web Of Science Queries

The set of works $\mathcal{W}$ is extended by all works receiving at least one citation by any work in $\mathcal{W}$. An edge (w_i, w_j) in the corresponding graph is assigned a length proportional to the inverse of common references of w_i and w_j, excluding self-edges. Note that we are only interested in proximities within the original set $\mathcal{W}$; all added works not already contained in $\mathcal{W}$ are not included in the visualization, but only serve to determine the strength of bibliographic coupling.

[1] http://scientific.thomson.com/products/wos/

[2] http://vlado.fmf.uni-lj.si/pub/networks/data/WoS/SN5.zip

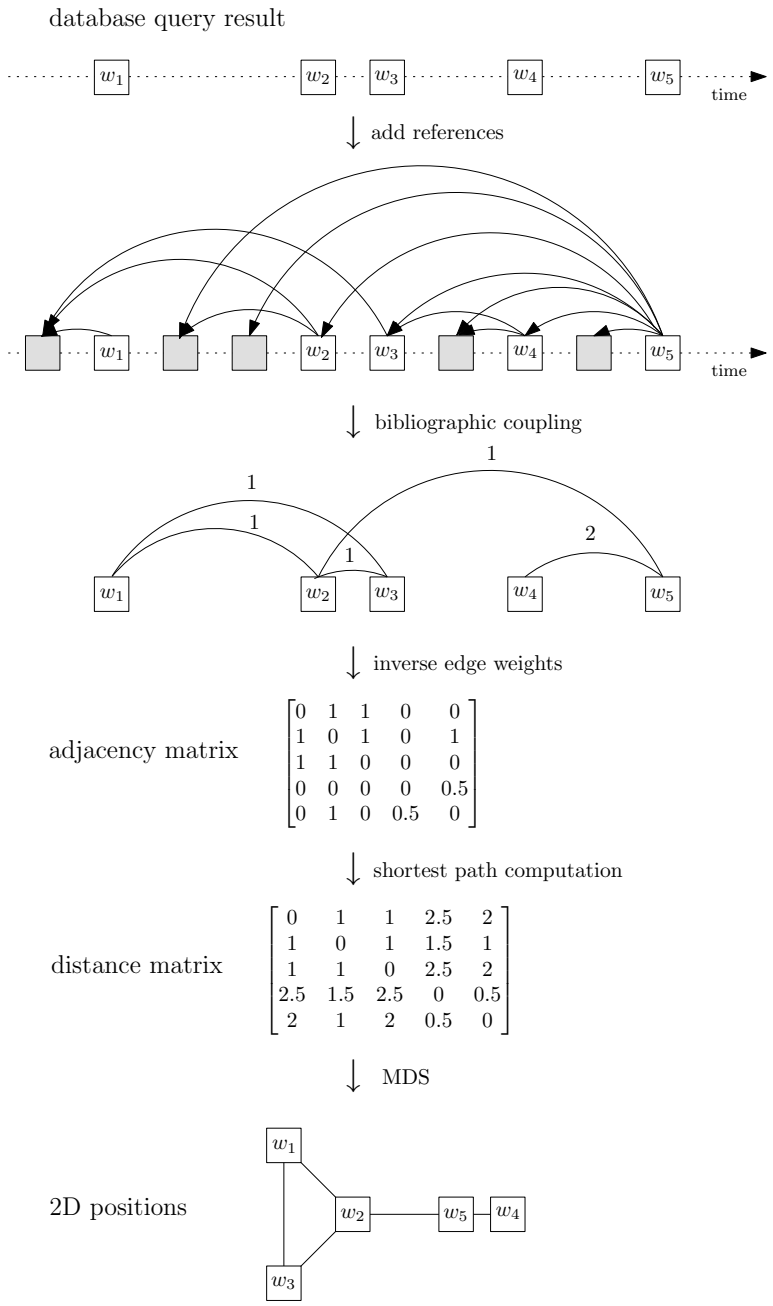


Figure 9.2: Basic workflow of the layout for a query result set $\mathcal{W} = \{w_1, \ldots, w_5\}$. The bibliographic coupling graph is generated by adding all cited works not already contained in $\mathcal{W}$ (gray). Using the distances in the resulting weighted bibliographic coupling graph, multidimensional scaling generates 2D positions.

The graph-theoretic distances in the resulting weighted graph are then subjected to classical scaling, giving the desired two-dimensional positions. The basic workflow of this visualization is depicted in Figure 9.2.

9.2.2 Bibliographic Coupling Networks

From the *SN5* data set we have generated and visualized networks describing bibliographic couplings in different domains: Works, authors, and journals. The corresponding networks were obtained after computing the inverse relative frequency and discarding all edges with a relative bibliographic coupling frequency below specific threshold values, which are adapted to the domain of the bibliographic objects. The following table gives the sizes of the largest connected components in 2007, for which we have produced visualizations:

domain	number of nodes	number of edges	threshold
works	5 463	87 528	0.25
authors	10 239	336 777	0.20
journals	1 673	60 263	0.15

9.2.3 Visualization

In our visualizations we use several graphical properties of bibliographic coupling edges. A striking spatial impression of visual clusterings is created by displaying and rendering edges after their strength, rendering the strongest couplings last and with thicker and darker edges. The layouts were generated using Pivot MDS with the number of pivots set to $k = 200$.

It is important to note that all presented visualizations of citations and the resulting bibliographic coupling are related to the *SN5* data set, since only those citations are considered. Therefore the observations about works, authors, and journals made on the basis of these visualizations are specific to this particular context, and an analysis and visualization based on other queries to the data base may lead to substantially different results.

Proximity of works Figure 9.3 shows the bibliographic coupling of works in 2007. The central component appears to be split into a left cluster of sociological articles, and a cluster on the right, with works originating in health sciences and psychology. These two clusters are connected by a bridge of works, which seem to be, not surprisingly, meta-studies of social network analysis methods from different fields, or works from social psychology. It is interesting to observe the small cluster in the lower left, which mainly contains physicists' works about network analysis.

The evolution of this bibliographic coupling network since 1990 is displayed in Figure 9.4. While the above-mentioned center has stayed stable over the years, in 2000 a new cluster nucleus of physicists' works is beginning to emerge. While the main component does not grow considerably, in 2005 the attached physicists' cluster has grown remarkably fast, indicating the increasing interest of the physics literature in social networks.

Proximity of authors The bibliographic coupling among the same set of works in 2007, but projected to authors, is shown in Figure 9.5. Similar to the proximity of works (Figure 9.3), there is an apparent separation into two main clusters of authors, which we attribute to two main directions "sociology" and "psychology and health". Again, a peripheral cluster of physicists can be observed, which is located in the lower right.

Proximity of journals Figure 9.6 displays the bibliographic coupling projected to journals. While there is once again a separation into groups quite similar to those in Figure 9.3 and Figure 9.5, the sociology journals have a noticeably peripheral position in the upper right part, indicating that articles published in the social sciences are frequently cited, but differ from other journals in the query result set significantly in their own citation behavior.

9.3 Conclusion

We have described a method for the visualization of proximity and distances in large bibliographic networks. It conveys various structural properties from different analytic perspectives, depending on the particular type of networks; the correspondingly defined proximities serve as a proxy for the similarity of the content of works and the general direction of interest of authors and journals.

Pivot MDS allows bibliographic data sets to be visualized even on a fairly large scale, and overcomes the limitations from which most other methods suffer. Moreover, it is easy to implement, since only basic graph traversals and matrix operations, but no sophisticated data structures, are required.

The presented visualizations of networks derived from bibliography data indicate that the method is successfully applied even to data sets of larger size. Note that there is no limitation to bibliographic networks, and the presented approach is easily carried over to other types of bipartite or multipartite affiliation networks. Therefore, we hope this application of classical scaling to the layout of large graphs is not only useful for the visual summary of large bibliography data, but also as a visualization building block in other fields of application.

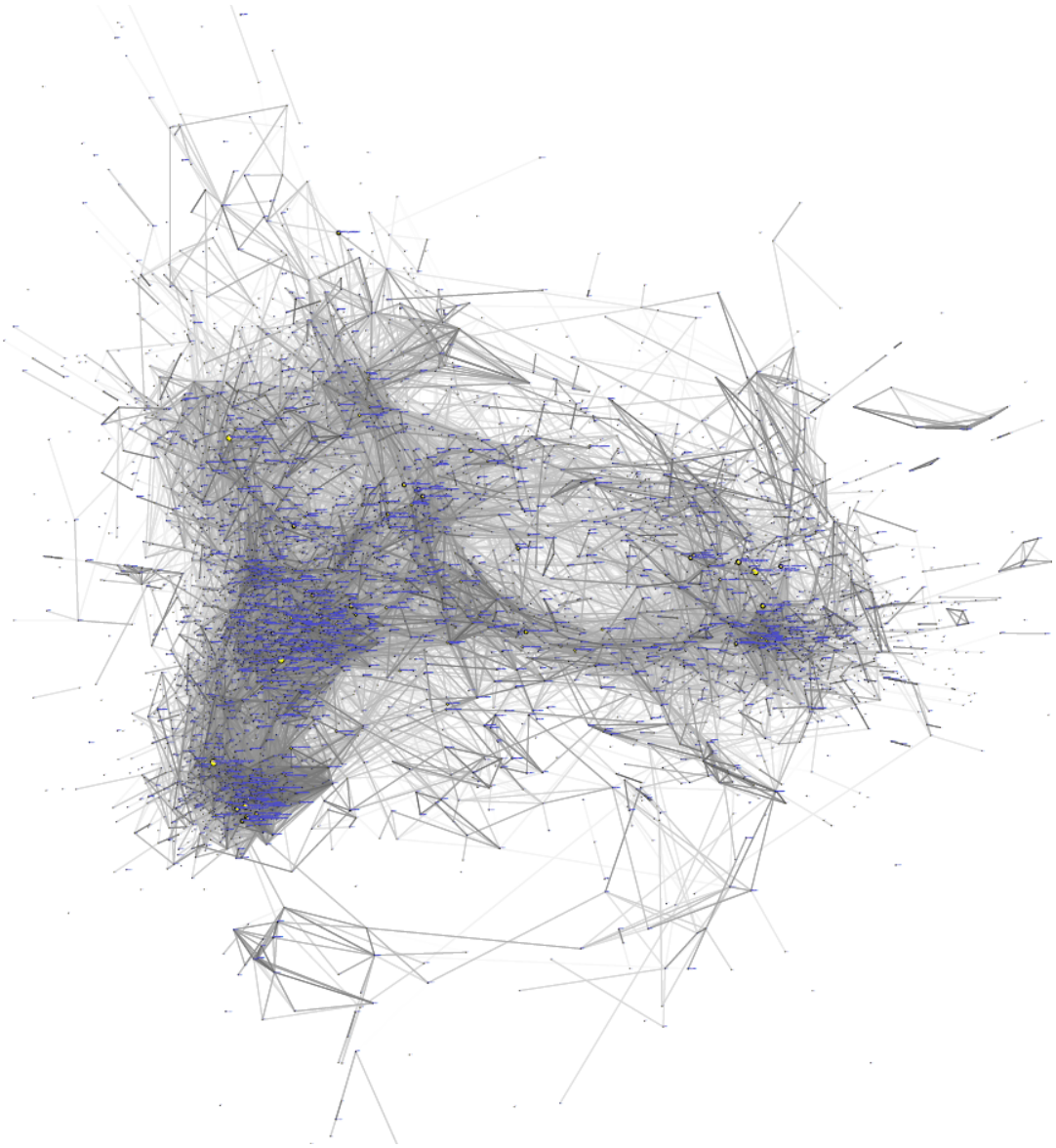
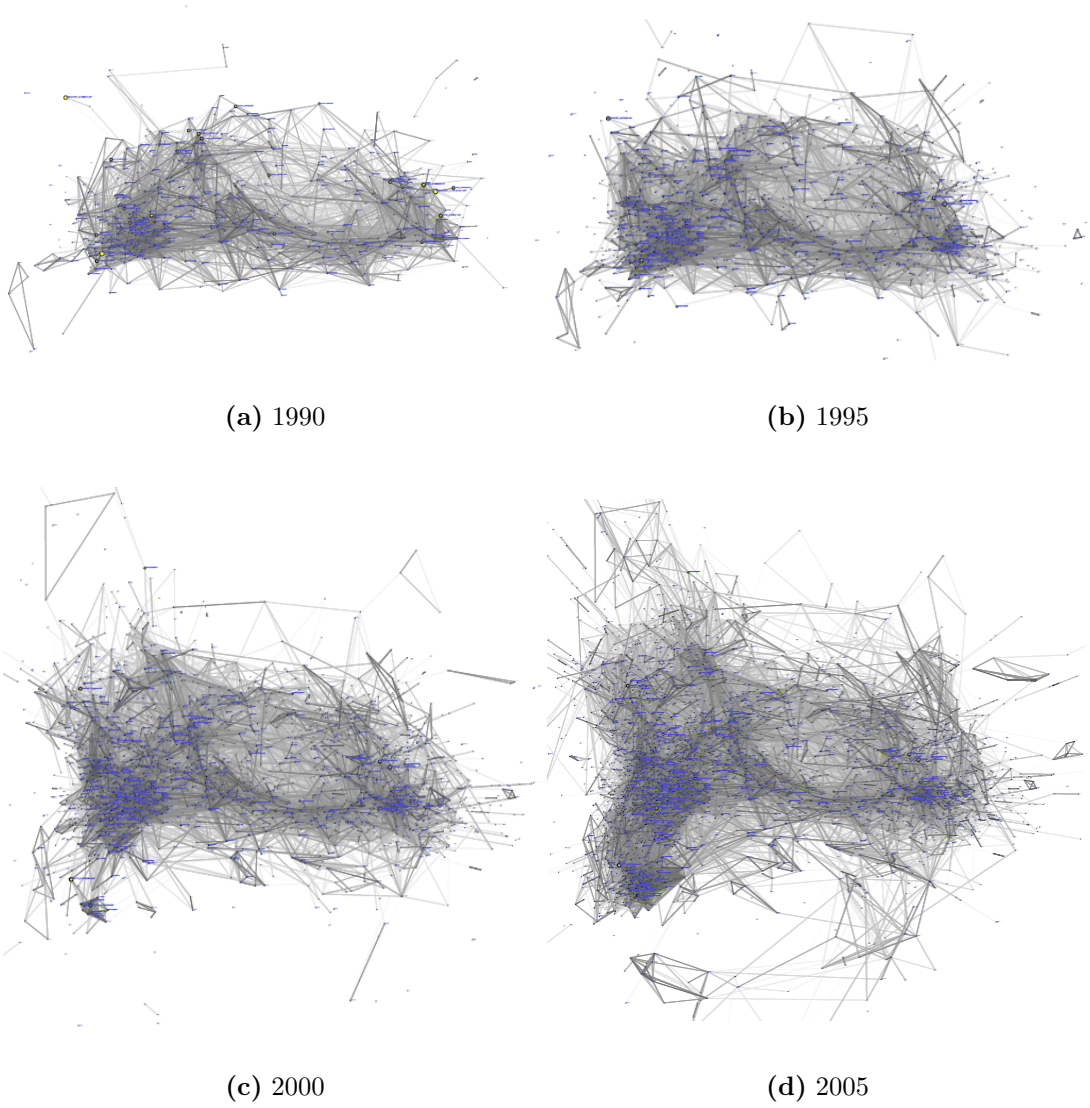


Figure 9.3: Bibliographic coupling network among works in the SN5 data set. The area of each work node is proportional to the number of works in which that work is cited.



(a) 1990 **(b)** 1995

(c) 2000 **(d)** 2005

Figure 9.4: Dynamic evolution of the bibliographic coupling graph of works, from 1990 to 2005. The layout is done with a flipbook approach, in which node positions reflect bibliographic proximity in 2007 and are fixed for all frames. In each frame the thickness and opacity of edges is proportional to the strength of bibliographic coupling at that time.

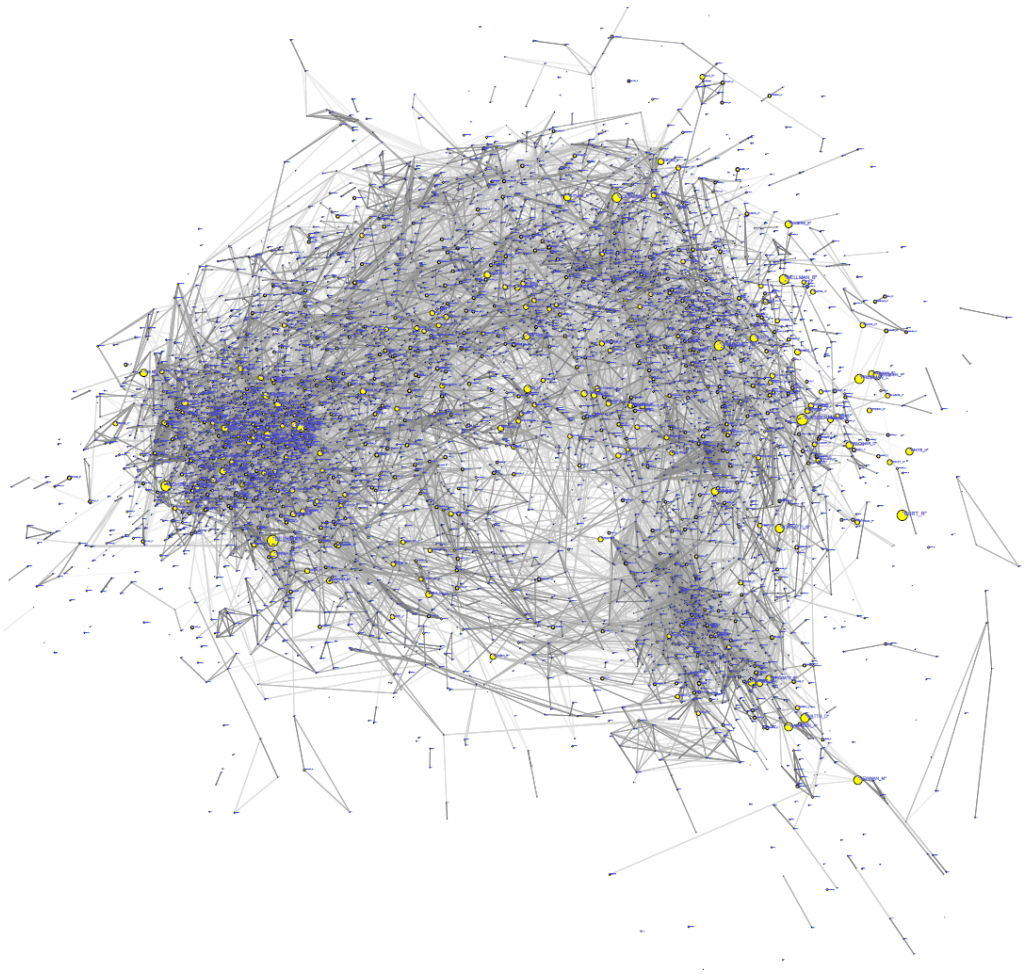


Figure 9.5: Bibliographic coupling network among authors in the SN5 data set. The area of each author node is proportional to the number of works in which that author is cited.

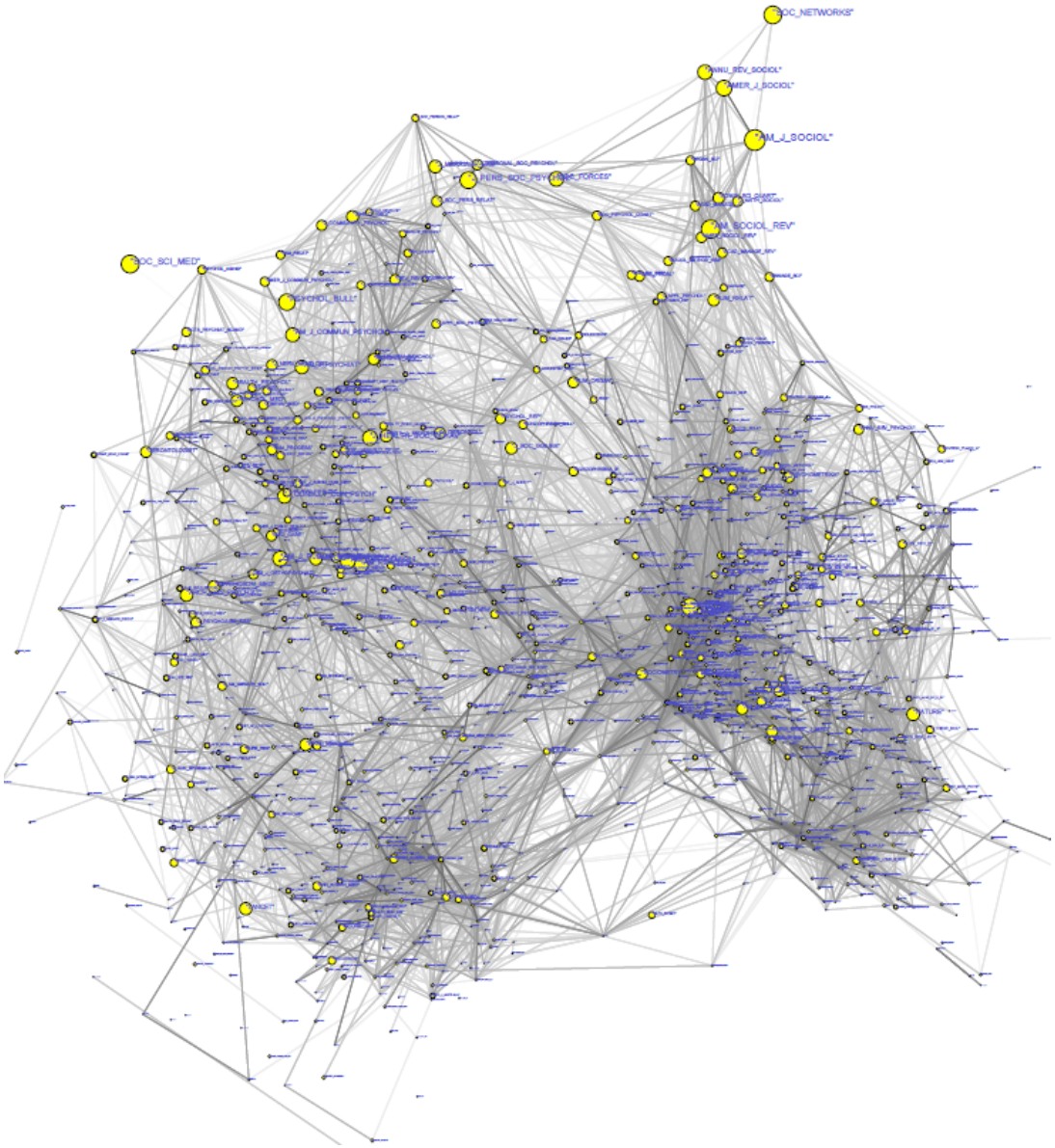


Figure 9.6: Bibliographic coupling network among journals in the SN5 data set. The area of each journal is proportional to the number of works in which that journal is cited.

Chapter 10

Conclusion

The main goal of this thesis was to provide a unified view on some models of multidimensional scaling and its applications to graph drawing. Many concepts and algorithmic techniques originating in other research disciplines (data analysis, statistics, psychometrics, geometry) are adapted to graph-theoretic settings and lead to attractive and effective methods for the visualization of graphs. The following list summarizes the principal research contributions of this thesis:

- Classical scaling, the earliest practical approach of reconstructing positions from distances, was applied to distances in graphs. Based on the hypothesis that the observed shortest-path distances are Euclidean distances, a provably optimal low-dimensional layout is obtained from certain eigenvectors of a matrix derived from the distances in the graph. The eigenvalues of this matrix can be used to state if this hypothesis is appropriate.

- We have presented a graph drawing method that is able to significantly speed up classical scaling. Based on computing only a very small part of the shortest-path distances, it yields a very fast layout algorithm with essentially the same quality. This speed-up technique allows classical scaling to be applied even to very large graphs, for which the original classical scaling and almost all other graph drawing methods are prohibitive due to their quadratic time and space complexity.

- The popular force-directed and energy-based graph drawing algorithms were put into the more general data-analytic framework of distance scaling.

- To impose additional constraints on the layout, the conventional stress measure was extended by an additional stress term as weak constraints. Minimizing the resulting objective function requires only minor modifications of existing MDS algorithms but widens the range of potential applications to graph layout problems.

- A novel method was introduced for the circular layout of directed graphs. Node positions are derived from certain eigenvectors of the skew-symmetric adjacency matrix associated with a directed graph; the directed edges are visualized as curves winding clockwise around the origin. The layout algorithm is easy to implement and runs in linear time.

- An extensive study provided experimental evidence for some common assumptions about the quality of MDS and related methods. The design of experiments is guided by hypotheses inspired from practical experience in the implementation of MDS and other graph drawing algorithms.

- Three applications illustrated the flexibility of multidimensional scaling and its scalability to larger problems: The joint visual analysis of importance and hierarchy in large software systems is based on a modification of the input distances; additional aesthetic criteria are explicitly formulated by engineering the weights in the objective functions; the application to the visual summary of large bibliographic data sets encompasses an advanced graph model and illustrates the scalability of MDS based methods to large problems.

We believe that the simplicity and elegance of multidimensional scaling are able to put forward its integration as a building block into software for graph visualization and network analysis. MDS has turned out to be an effective method for the automatic visualization of graphs, since, on the algorithmic side, it has proven to be more robust and reliable than the traditional force-based approaches; on the conceptual side, it is particularly attractive because it allows aesthetic criteria to be integrated explicitly in its objective functions, which, in turn, facilitates interpretation.

Some new possible lines of research have been identified, which could not be followed systematically but appear to be promising in their own right and may give rise to new graph drawing methods:

- The speedup of distance scaling with a suitable stress sparsification scheme and performance guarantees for MDS speedup methods.

- Characterizing the spectral properties of graph-theoretical distances and the use of alternative graph distances for the layout.

- Adapting further MDS methodologies to graph drawing, e.g. non-metric scaling or multi-mode and multi-way models.

List of Figures

3.1 Layout matrix of the first five dimensions of classical scaling . . . 24
3.2 Extremal regions of the spectra of ten example graphs 26
3.3 Pivoting strategies Random vs. Maxmin 28
3.4 Progressive Pivot MDS 32
3.5 Drawing very large graphs with Pivot MDS 33
3.6 Pivot MDS vs. Landmark MDS 35
3.7 The connection between PCA and MDS 37
3.8 HDE vs. Pivot MDS 40

4.1 MDS layouts with several weighting schemes 44
4.2 Majorization steps 46
4.3 Confirmed and unconfirmed edges 56

5.1 Drawings and Shepard plots I 64
5.2 Drawings and Shepard plots II 65
5.3 Initialization strategies 68
5.4 Drawings with different initializations 69
5.5 Pivot MDS vs. Landmark MDS 70
5.6 Sparse stress 72

6.1 Skew-symmetry represented by signed triangle areas 77
6.2 Cases of skew-symmetry in a bimension. 79
6.3 Clockwise drawings with straight lines and curves 82
6.4 All bimensions of a small graph 84
6.5 Drawing curved edges using splines with control points. 85
6.6 Rotation effect of the power iteration in skew-symmetric scaling . 86
6.7 Tournament graphs in European football leagues 87
6.8 Polar transform of a hierarchical tournament 89

7.1 PageRank and TrustRank as vertical coordinates 98
7.2 Java compile-time dependencies I 102
7.3 Java compile-time dependencies II 103

7.4 Dependencies in a Microsoft software project (PageRank) 106
7.5 Dependencies in a Microsoft software project (TrustRank) 107

8.1 Radial layouts of a social network 111
8.2 Radial layouts of the karate club network 114
8.3 Centrality layouts of the karate club network 116
8.4 Travel time layouts for the London Tube graph 119

9.1 Data model for bibliographic data 123
9.2 Workflow for visualiazing a query 127
9.3 Proximity of works 130
9.4 Dynamic evolution of proximity among works 131
9.5 Proximity of authors 132
9.6 Proximity of journals 133

List of Algorithms

1 Classical scaling 16
2 Power iteration for spectral decomposition 20
3 Construction of a pivot set 29
4 Pivot MDS 31
5 Landmark MDS 36
6 Stress majorization 48
7 Graph drawing by stress majorization 52
8 Stress majorization with weak constraints 54
9 Decomposing a skew-symmetric matrix 80
10 Clockwise drawing of a directed graph 88
11 Radial layout 117

Bibliography

Abelson, S. J. and Messick, R. P. (1956). The additive constant problem in multidimensional scaling. *Psychometrika*, 21:1–15. (Cited on page 23.)

Alfakih, A. Y. (2000). Graph rigidity via euclidean distance matrices. *Linear Algebra and its Applications*, 310:149–165. (Cited on page 25.)

Ambrosi, K. and Hansohm, J. (1987). Ein dynamischer Ansatz zur Repräsentation von Objekten. In *Operations Research Proceedings 1986*, pages 425–431. (Cited on page 55.)

Anthonisse, J. M. (1971). The rush in a directed graph. Technical report, Stichting Mathematisch Centrum, Amsterdam. (Cited on page 115.)

Bachmaier, C., Brandenburg, F.-J., Brunner, W., and Lovász, G. (2009). Cyclic leveling of directed graphs. In *Proceedings of the 15th International Symposium on Graph Drawing (GD '08)*, volume 5417 of *Lecture Notes in Computer Science*, pages 348–359. (Cited on pages 81 and 89.)

Bachmaier, C., Brandenburg, F.-J., and Forster, M. (2005). Radial level planarity testing and embedding in linear time. *Journal of Graph Algorithms and Applications*, 9(1):53–97. (Cited on page 109.)

Bastert, O. and Matuszewski, C. (2001). Layered drawings of digraphs. In Kaufmann, M. and Wagner, D., editors, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 87–120. Springer. (Cited on page 94.)

Bay, T. G. and Pauls, K. (2005). Reuse frequency as metric for dependency resolver selection. In *Proceedings of the 3rd International Working Conference on Component Deployment*, pages 164–176. (Cited on page 94.)

Bellman, R. E. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87—90. (Cited on page 22.)

Bengio, Y., Paiement, J.-F., Vincent, P., Delalleau, O., Le Roux, N., and Ouimet, M. (2004). Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, pages 307–311. (Cited on page 25.)

Bevan, J. and Whitehead, E. J. (2003). Identification of software instabilities. In *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE 2003)*, pages 134–145. (Cited on page 94.)

Boisvert, R., Pozo, R., Remington, K., Barrett, R., and Dongarra, J. (1997). The matrix market: A web resource for test matrix collections. In *Quality of Numerical Software, Assessment and Enhancement*, pages 125–137. Chapman Hall. (Cited on page 60.)

Borg, I. and Groenen, P. (2005). *Modern Multidimensional Scaling*. Springer, 2nd edition. (Cited on pages 13 and 52.)

Borg, I. and Lingoes, J. C. (1980). A model and algorithm for multidimensional scaling with external constraints on the distances. *Psychometrika*, 45(1):25–38. (Cited on pages 52, 53, 112, and 114.)

Brandenburg, F.-J., Himsolt, M., and Rohrer, C. (1996). An experimental comparison of force-directed and randomized graph drawing algorithms. In North, S. R., editor, *Proceedings of the International Symposium on Graph Drawing (GD '96)*, volume 1027 of *Lecture Notes in Computer Science*, pages 76–87. (Cited on page 58.)

Brandes, U. (2001). Drawing on physical analogies. In Kaufmann, M. and Wagner, D., editors, *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*, pages 71–86. Springer. (Cited on pages 49 and 57.)

Brandes, U. and Cornelsen, S. (2003). Visual ranking of link structures. *Journal of Graph Algorithms and Applications*, 7(2):181–201. (Cited on pages 94 and 104.)

Brandes, U., Kenis, P., and Wagner, D. (2003). Communicating centrality in policy network drawings. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):241–253. (Cited on pages 94, 110, and 115.)

Brandes, U. and Pich, C. (2007). Eigensolver methods for progressive multidimensional scaling of large data. In *Proceedings of the 16th International Symposium on Graph Drawing (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 42–53. (Cited on pages ii, 4, 21, 62, and 67.)

Brandes, U. and Pich, C. (2009a). An experimental study on distance-based graph drawing. In Tollis, I. G. and Patrignani, M., editors, *Proceedings of the 16th International Symposium in Graph Drawing (GD'08)*, volume 5417 of *Lecture Notes in Computer Science*, pages 218–229. (Cited on pages ii and 4.)

Brandes, U. and Pich, C. (2009b). More flexible radial layout. In *Proceedings of the 17th International Symposium in Graph Drawing (GD'09)*. To appear. (Cited on pages ii and 4.)

Brandes, U. and Willhalm, T. (2002). Visualization of bibliographic networks with a reshaped landscape metaphor. In *Proceedings of the Joint Eurographics/IEEE TCVG Symposium on Visualization*, pages 159–164. (Cited on pages 94, 104, and 121.)

Brinkmeier, M. and Schank, T. (2005). Network statistics. In Brandes, U. and Erlebach, T., editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, pages 293–317. Springer. (Cited on page 23.)

Buckley, F. and Harary, F. (1989). *Distances in Graphs*. Addison-Wesley. (Cited on page 23.)

Buja, A. and Swayne, D. F. (2002). Visualization methodology for multidimensional scaling. *Journal of Classification*, 19:7–43. (Cited on pages 45, 57, and 58.)

Buja, A., Swayne, D. F., Littmann, M. L., Dean, N., and Hofmann, H. (2001). XGvis: Interactive data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*. To appear. (Cited on page 21.)

Burges, C. J. C. (2004). Geometric methods for feature extraction and dimensional reduction. Technical report, Microsoft Research. (Cited on page 25.)

Carden, T. (2005). Travel time tube map. `http://www.tom-carden.co.uk/p5/tube_map_travel_times/applet/`. (Cited on page 118.)

Carmel, L., Harel, D., and Koren, Y. (2004). Combining hierarchy and energy for drawing directed graphs. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):46–57. (Cited on pages 80 and 94.)

Carroll, J. and Chang, J. (1972). Idioscal (individual differences in orientation scaling): A generalization of indscal allowing idiosyncratic reference systems as well as an analytic approximation to indscal. Paper presented at the Psychometric Society Meeting. (Cited on page 18.)

Cayley, A. (1841). On a theorem in the geometry of position. *Cambridge Mathematical Journal*, 2:267–271. (Cited on page 14.)

Chalmers, M. (1996). A linear iteration time layout algorithm for visualizing high-dimensional data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 1996)*, pages 127–132. IEEE. (Cited on page 25.)

Chen, T. T. and Hsieh, L. C. (2007). On visualization of cocitation networks. In *Proceedings of the 11th International Conference Information Visualization (IV '07)*, pages 470–475. (Cited on page 121.)

Çivril, A., Magdon-Ismail, M., and Bocek-Rivele, E. (2006). SDE: Graph drawing using spectral distance embedding. In Healy, P., editor, *Proceedings of the 13th International Symposium on Graph Drawing (GD '05)*, volume 3843 of *Lecture Notes in Computer Science*, pages 512–513. (Cited on page 21.)

Çivril, A., Magdon-Ismail, M., and Bocek-Rivele, E. (2007). SSDE: Fast graph drawing using sampled spectral distance embedding. In Kaufmann, M. and Wagner, D., editors, *Proceedings of the 14th International Symposium on Graph Drawing (GD '06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 30–41. (Cited on page 21.)

Cohen, J. D. (1997). Drawing graphs to convey proximity. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229. (Cited on pages 45 and 48.)

Consens, M., Mendelzon, A., and Ryman, A. (1992). Visualizing and querying software structures. In *Proceedings of the 14th International Conference on Software Engineering*, pages 138–156. (Cited on page 95.)

Coombs, C. H. (1950). Psychological scaling without a unit of measurement. *Psychological Review*, 57:148–158. (Cited on page 37.)

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, 2nd edition. (Cited on pages 5 and 22.)

Cox, T. F. and Cox, M. A. A. (2001). *Multidimensional Scaling*. CRC/Chapman and Hall, 2nd edition. (Cited on page 13.)

Critchley, F. (1988). On certain linear mappings between inner-product and squared-distance matrices. *Linear Algebra and Applications*, 105:91–107. (Cited on page 15.)

Davidson, R. and Harel, D. (1996). Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331. (Cited on page 110.)

de Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. In Barra, J. R., Brodeau, F., Romier, G., and van Cutsem, B., editors, *Recent*

Developments in Statistics, pages 133–145. Amsterdam: North-Holland. (Cited on pages 45, 47, 48, and 61.)

de Leeuw, J. (1988). Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180. (Cited on page 48.)

de Leeuw, J. and Heiser, W. J. (1980). Multidimensional scaling with restrictions on the configuration. In Krishnaia, P., editor, *Multivariate Analysis*, pages 501–522. North-Holland. (Cited on page 53.)

de Leeuw, J. and Michailidis, G. (2000). Graph layout techniques and multidimensional analysis. In *IMS Lecture Notes-Monograph Series*, pages 219–248. Thomas S. Ferguson and F.T. Bruss and L. Le Cam. (Cited on pages 49 and 51.)

de Silva, V. and Tenenbaum, J. (2003). Global versus local methods in nonlinear dimensionality reduction. In Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15, pages 721–728. (Cited on page 33.)

de Silva, V. and Tenenbaum, J. B. (2004). Sparse multidimensional scaling using landmark points. Technical report, Stanford University. (Cited on page 62.)

DeLine, R., Khella, A., Czerwinski, M., and Robertson, G. (2005). Towards understanding programs through wear-based filtering. In *Proceedings of the 2005 ACM symposium on Software Visualization (SoftVis 2005)*, pages 183–192. (Cited on page 93.)

DeSarbo, W. S. and Rao, V. R. (1984). Genfold2: A set of models and algorithms for the general unfolding analysis of preference/dominance data. *Journal of Classification*, 1:147–186. (Cited on page 45.)

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall. (Cited on page 50.)

Diestel, R. (2006). *Graph Theory*. Number 173 in Graduate Texts in Mathematics. Springer, 3rd edition. (Cited on page 5.)

Dijkstra, E. W. (1959). A note on two problems in the connexion with graphs. *Numerische Mathematik*, 1:269–271. (Cited on page 22.)

Duff, I. S., Grimes, R. G., and Lewis, J. G. (1989). Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14. (Cited on page 60.)

Duncan, C. A., Klau, G., Kobourov, S. G., and Sander, G. (2007). Graph-drawing contest report. In *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 42–53. (Cited on page 100.)

Dwyer, T. and Koren, Y. (2005). Dig-cola: Directed graph layout through constrained energy minimizatio. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2005)*, pages 65–72. (Cited on page 53.)

Dwyer, T., Koren, Y., and Marriott, K. (2005). Stress majorization with orthogonal ordering constraints. In *Proceedings of 13th International Symposium on Graph Drawing*, pages 141–152. (Cited on page 94.)

Dwyer, T., Koren, Y., and Marriott, K. (2006a). Drawing directed graphs using quadratic programming. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):536–548. (Cited on page 97.)

Dwyer, T., Koren, Y., and Marriott, K. (2006b). Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):821–828. (Cited on pages 53 and 94.)

Dwyer, T., Marriott, K., and Wybrow, M. (2009). Topology preserving constrained graph layout. In Tollis, I. G. and Patrignani, M., editors, *Proceedings of the 16th International Symposium in Graph Drawing (GD'08)*, volume 5417 of *Lecture Notes in Computer Science*, pages 230–241. (Cited on page 110.)

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160. (Cited on page 50.)

Eades, P. and Wormald, N. C. (1990). Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28(2):111–134. (Cited on page 61.)

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218. (Cited on page 17.)

Faloutsos, C. and Lin, K.-I. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference*, pages 163–174. (Cited on page 25.)

Ferrante, J., Ottenstein, K. J., and Warren, J. D. (1987). The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9:319–349. (Cited on page 94.)

Fleischer, D. (2007). *Theory and Applications of the Laplacian.* PhD thesis, Universität Konstanz. (Cited on page 50.)

Floyd, R. W. (1962). Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345. (Cited on page 22.)

Ford, L. R. (1956). Network flow theory. Technical report, The Rand Corporation, Santa Monica. (Cited on page 22.)

Fouss, F., Pirotte, A., Renders, J., and Saerens, M. (2005). A novel way of computing dissimilarities between nodes of a graph, with application to collaborative recommendation. In *Proceedings of the International Conference on Web Intelligence*, pages 550–556. (Cited on page 50.)

Fredman, M. L. and Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615. (Cited on page 22.)

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41. (Cited on page 115.)

Freeman, L. C. (2000). Visualizing social networks. *Journal of Social Structure*, 1. (Cited on page 57.)

Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software, Practice and Experience*, 21(11):1129–1164. (Cited on page 50.)

Gajer, P., Goodrich, M. T., and Kobourov, S. G. (2000). A multi-dimensional approach to force-directed layouts of large graphs. In *Proceedings of the 8th International Symposium on Graph Drawing (GD '00)*, volume 1984 of *Lecture Notes in Computer Science*, pages 211–221. (Cited on page 51.)

Gajer, P. and Kobourov, S. (2001). GRIP – Graph drawing with intelligent placement. In *Proceedings of the 8th International Symposium on Graph Drawing (GD '00)*, volume 1984 of *Lecture Notes in Computer Science*, pages 222–228. (Cited on pages 51 and 61.)

Ganapathy, M. K. and Lodha, S. P. (2004). On minimum circular arrangement. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS '04)*, pages 394–405. (Cited on pages 81 and 90.)

Gansner, E. R. and Hu, Y. (2009). Efficient node overlap removal using a proximity stress model. In Tollis, I. G. and Patrignani, M., editors, *Proceedings of*

the 16th International Symposium in Graph Drawing (GD'08), volume 5417 of *Lecture Notes in Computer Science*, pages 206–217. (Cited on pages 55 and 110.)

Gansner, E. R., Koren, Y., and North, S. (2004). Graph drawing by stress majorization. In Pach, J., editor, *Proceedings of the 12th International Symposium on Graph Drawing (GD '04)*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. (Cited on pages 47, 48, 51, 57, and 67.)

Garland, K. (1994). *Mr. Beck's Underground Map*. Capital Transport Publishers. (Cited on page 117.)

Gauß, C. F. (1844/1910). Untersuchungen über Gegenstände der höheren Geodäsie. In *Ostwalds Klassiker der exakten Wissenschaften Nr. 177*. Wilhelm Engelmann. (Cited on page 14.)

Gold, E. (1973). Metric unfolding: Data requirement for unique solution and clarification of Schönemann's algorithm. *Psychometrika*, 38(4):555–569. (Cited on page 38.)

Golub, G. H. and van Loan, C. F. (1996). *Matrix Computations*. The Johns Hopkins University Press, 3rd edition. (Cited on pages 5 and 21.)

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53(3–4):325–338. (Cited on pages 14, 18, and 37.)

Gower, J. C. (1968). Adding a point to vector diagrams in multivariate analysis. *Biometrika*, 55(3):582–585. (Cited on page 34.)

Gower, J. C. (1977). The analysis of asymmetry and orthogonality. *Recent Developments in Statistics*, pages 109–123. (Cited on page 75.)

Gower, J. C. and Constantine, A. G. (1978). Graphical representation of asymmetric matrices. *Applied Statistics*, 27:297–304. (Cited on pages 76 and 82.)

Gower, J. C. and Digby, P. G. N. (1981). Expressing complex relationships in two dimensions. In Barnett, V., editor, *Interpreting Multvariate Data*, chapter 6, pages 83–118. John Wiley & Sons. (Cited on page 75.)

Gower, J. C. and Legendre, P. (1986). Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48. (Cited on page 124.)

Graef, J. and Spence, I. (1979). Using distance information in the design of large multidimensional scaling experiments. *Psychological Bulletin*, 486:60–66. (Cited on page 43.)

Guttman, L. (1968). A general nonmetric techniqe for finding the smallest coordinate space for a configuration of points. *Psychometrika*, 33:469–506. (Cited on pages 45 and 47.)

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. (2004). Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 576–587. (Cited on page 97.)

Hachul, S. and Jünger, M. (2004). Drawing large graphs with a potential-field-based multilevel algorithm. In Pach, J., editor, *Proceedings of the 12th International Symposium on Graph Drawing (GD '04)*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. (Cited on page 51.)

Hachul, S. and Jünger, M. (2006). An experimental comparison of fast algorithms for drawing general large graphs. In Healy, P., editor, *Proceedings of the 13th International Symposium on Graph Drawing (GD '05)*, volume 3843 of *Lecture Notes in Computer Science*, pages 235–250. (Cited on pages 33, 39, 58, and 61.)

Hadany, R. and Harel, D. (2001). A multi-scale method for drawing graphs nicely. *Discrete Applied Mathematics*, 113:3–21. (Cited on page 51.)

Hall, K. M. (1970). An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229. (Cited on pages 21 and 49.)

Harary, F. and Melter, R. A. (1976). On the metric dimension of a graph. *Ars Combinatoria*, 15(2):191–195. (Cited on page 25.)

Harary, F. and Moser, L. (1966). The theory of round robin tournaments. *The American Mathematical Monthly*, 73:231–246. (Cited on page 86.)

Harel, D. and Koren, Y. (2002a). A fast multi-scale method for drawing large graphs. *Journal of Graph Algorithms and Applications*, 6:179–202. (Cited on page 51.)

Harel, D. and Koren, Y. (2002b). Graph drawing by high-dimensional embedding. In Kobourov, S. G. and Goodrich, M. T., editors, *Proceedings of the 11th International Symposium on Graph Drawing (GD '02)*, volume 2528 of *Lecture Notes in Computer Science*, pages 207–219. (Cited on pages 38 and 61.)

Harshman, R. A. and Lundy, M. E. (1990). Multidimensional analysis of preference structures. In de Fontenay, A., Shugard, M. H., and Shibley, D. S., editors, *Telecommunications Demand Modelling*, pages 185–204. Elsevier. (Cited on pages 74 and 77.)

Haveliwala, T. (2003). Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796. (Cited on page 100.)

He, Y. and Hui, S. C. (2001). Mining a web citation database for author co-citation analysis. *Information Processing and Management*, 38(4):491–508. (Cited on page 124.)

Heiser, W. J. (1998). Multidimensional scaling with least absolute residuals. In Bock, H. H., editor, *Classification and Related Methods*. North-Holland. (Cited on pages 42 and 45.)

Heiser, W. J. and Meulman, J. (1983a). Analyzing rectangular tables by joint and constrained multidimensional scaling. *Econometrics*, 22:139–167. (Cited on pages 34 and 38.)

Heiser, W. J. and Meulman, J. (1983b). Constrained multidimensional scaling, including confirmation. *Applied Psychological Measurement*, 7(4):381–404. (Cited on page 52.)

Hochbaum, D. S. and Shmoys, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations research*, 10(22):180–184. (Cited on page 27.)

Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748. (Cited on page 95.)

Hong, S.-H., Merrick, D., and do Nascimiento, H. A. D. (2004). The metro map layout problem. In *Proceedings of the 2004 Australasian symposium on Information Visualisation*, ACM International Conference Proceeding Series, pages 91–100. (Cited on page 117.)

Hosobe, H. (2004). A high-dimensional approach to interactive graph visualization. In *Proceedings of the ACM Symposium on Applied Computing (SAC 2004)*, pages 1253–1257. (Cited on page 21.)

Hosobe, H. (2005). An extended high-dimensional method for interactive graph drawing. In *Proceedings of the Asia Pacific Symposium on Information Visualisation (APVIS 2005)*, volume 109 of *ACM International Conference Proceeding Series*, pages 15–20. Australian Computer Society. (Cited on page 21.)

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 498–520. (Cited on page 18.)

Indyk, P. (2001). Tutorial: Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS '01)*, pages 10–33. (Cited on page 25.)

Ito, T., Tashiro, K., Muta, S., Ozawa, R., Chiba, T., Nishizawa, M., Yamamoto, K., Kuhara, S., and Sakaki, Y. (2000). Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proceedings of the National Academy of Sciences (PNAS)*, 97(3):1143–1147. (Cited on page 60.)

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579. (Cited on page 125.)

Jeh, G. and Widom, J. (2003). Scaling personalized web search. In *Proceedings of the 12th International Conference on World Wide Web*, pages 271–279. (Cited on page 100.)

Joliffe, I. T. (1986). *Principal Component Analysis*. Springer. (Cited on page 36.)

Jourdan, F. and Melançon, G. (2004). Multiscale hybrid MDS. In *Proceedings of the 8th IEEE International Conference on Information Visualization (IV '04)*, pages 388–393. IEEE. (Cited on page 25.)

Jungmayr, S. (2002). Testability measurement and software dependencies. In *Proceedings of the 12th International Workshop on Software Measurement*, pages 179–202. (Cited on page 94.)

Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15. (Cited on pages 51, 57, and 61.)

Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H. (2003). Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University. (Cited on page 100.)

Kaufmann, M. and Wagner, D. (2001). *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag. (Cited on page 82.)

Kessler, M. M. (1963). Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25. (Cited on page 124.)

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632. (Cited on page 85.)

Koren, Y. (2003). On spectral graph drawing. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON '03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 496–508. Springer-Verlag. (Cited on pages 21 and 49.)

Koren, Y. (2005). Drawing graphs by eigenvectors: Theory and practice. *Computers and Mathematics with Applications*, 49(11–12):1867–1888. (Cited on page 104.)

Koren, Y. and Çivril, A. (2009). The binary stress model for graph drawing. In *Proceedings of the 16th International Symposium in Graph Drawing (GD'08)*, volume 5417 of *Lecture Notes in Computer Science*, pages 193–205. (Cited on page 110.)

Koren, Y. and Harel, D. (2005). One-dimensional layout optimization, with applications to graph drawing by axis separation. *Computational Geometry: Theory and Applications*, 32:115–138. (Cited on pages 21, 39, and 94.)

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27. (Cited on pages 13, 43, 45, and 61.)

Kruskal, J. B. (1971). Comments on "a nonlinear mapping for data structure analysis" (letter to the editor). *IEEE Transactions on Computers*, C-20(12):1614. (Cited on page 45.)

Kruskal, J. B. and Hart, R. E. (1966). A geometric interpretation of diagnostic data from a digital machine: Based on a study of the Morris, Illinois, Electronic Central Office. *Bell System Technical Journal*, 45(8):1299–1338. (Cited on page 25.)

Kruskal, J. B. and Seery, J. B. (1980). Designing network diagrams. In *Proceedings of the First General Conference on Social Graphics*, pages 22–50. (Cited on pages 21, 48, and 57.)

Krzanowski, W. J. (2006). Sensitivity in metric scaling and analysis of distance. *Biometrics*, 62:239–244. (Cited on page 18.)

Langville, A. N. and Meyer, C. D. (2005). A survey of eigenvector methods of web information retrieval. *The SIAM Review*, 47(1):135–161. (Cited on page 100.)

Lanza, M. and Ducasse, S. (2003). Polymetric views: A lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795. (Cited on page 95.)

Lewis, R. M. and Trosset, M. W. (2006). Sensitivity analysis of the strain criterion for multidimensional scaling. *Computational Statistics and Data Analysis*, 50(1):135–153. (Cited on page 18.)

Liberatore, V. (2004). Circular arrangements and cyclic broadcast scheduling. *Journal of Algorithms*, 51(2):185–215. (Cited on pages 81 and 90.)

Linial, N., London, E., and Rabinovich, Y. (1995). The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245. (Cited on page 25.)

Lipp, C. (2000). Political and revolutionary culture in a German town 1830–1850. *History and Computing*, 12:71–81. (Cited on page 60.)

Malone, S. W., Tarazaga, P., and Trosset, M. W. (2002). Better initial configurations for metric multidimensional scaling. *Computational Statistics and Data Analysis*, 41(1):143–156. (Cited on pages 48 and 61.)

Mardia, K. V. (1978). Some properties of classical multi-dimensional scaling. *Communications in Statistics – Theory and Methods*, 7(13):1233–1241. (Cited on page 18.)

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press. (Cited on page 13.)

McGee, V. E. (1966). The multidimensional scaling of "elastic" distances. *The British Journal of Mathematical and Statistical Psychology*, 19:181–196. (Cited on pages 45, 51, and 61.)

Mead, A. (1992). Review of the development of multidimensional scaling methods. *The Statistician*, 41:27–39. (Cited on page 14.)

Menger, K. (1928). Untersuchungen über allgemeine Metrik. *Annalen der Mathematik*, 100:75–163. (Cited on page 14.)

Meyer, C. D. (2001). *Matrix Analysis and Applied Linear Algebra*. SIAM, 3rd edition. (Cited on page 5.)

Mihalić, Z., Veljan, D., Amić, D., Nikolić, S., Plavšić, D., and Trinajstić, N. (1992). The distance matrix in chemistry. *Journal of Mathematical Chemistry*, 11:223–258. (Cited on page 25.)

Morrison, A. and Chalmers, M. (2003). Improving hybrid MDS with pivot-based searching. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2003)*, pages 85–90. IEEE. (Cited on page 25.)

Morrison, A., Ross, G., and Chalmers, M. (2002). A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002)*, pages 152–158. IEEE. (Cited on page 25.)

Müller, H. A., Orgun, M. A., Tilley, S. R., and Uhl, J. S. (1993). A reverse-engineering approach to subsystem structure identification. *Journal of Software Maintenance: Research and Practice*, 5(4):181–204. (Cited on pages 93 and 95.)

Nagappan, N. and Ball, T. (2007). Using software dependencies and churn metrics to predict field failures: An empirical case study. In *Proceedings of the 5th International Symposium on Empirical Software Engineering*, pages 364–373. (Cited on page 94.)

Nakanishi, M. and Cooper, L. G. (2003). Metric unfolding revisited. Technical report, Department of Statistics, UCLA. (Cited on page 38.)

Neumann, P., Schlechtweg, S., and Carpendale, M. S. T. (2005). Arctrees: Visualizing relations in hierarchical data. In Brodlie, K. W., Duke, D. J., and Joy, K. I., editors, *Data Visualization 2005, Eurographics/IEEE VGTC Symposium on Visualization Symposium Proceedings*, pages 53–60. (Cited on page 95.)

Noack, A. (2007). Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480. (Cited on page 51.)

Northway, M. L. (1940). A method for depicting social relationships obtained by sociometric testing. *Sociometrics*, 3:144–150. (Cited on page 109.)

Orso, A., Sinha, S., and Harrold, M. J. (2004). Classifying data dependences in the presence of pointers for program comprehension, testing, and debugging. *ACM Transactions on Software Engineering and Methodology*, 13:199–239. (Cited on page 94.)

Paardekooper, M. H. C. (1971). An eigenvalue algorithm for skew-symmetric matrices. *Numerische Mathematik*, 17(3):189–202. (Cited on page 79.)

Padmanabhan, D., Desikan, P., Srivastava, J., and Riaz, K. (2005). Wicer: A weighted inter-cluster edge ranking for clustered graphs. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 522–528. (Cited on page 100.)

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University. (Cited on page 96.)

Pich, C. (2009). Drawing directed graphs clockwise. In *Proceedings of the 17th International Symposium in Graph Drawing (GD'09)*. To appear. (Cited on pages ii and 4.)

Pich, C., Nachmanson, L., and Robertson, G. (2008). Visual analysis of importance and grouping in software dependency graphs. In *Proceedings of the ACM Symposium on Software Visualization (SoftVis '08)*, pages 29–32. ACM. (Cited on pages ii and 4.)

Platt, J. C. (2004). FastMap, MetricMap, and Landmark MDS are all Nyström Algorithms. Technical report, Microsoft Research. (Cited on page 25.)

Podgurski, A. and Clarke, L. A. (1990). A formal model of program dependencies and its implications for software testing, debugging, and maintenance. *IEEE Transactions on Software Engineering*, 16(9):965–979. (Cited on page 94.)

Quinn, N. R. and Breuer, M. A. (1979). A force directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, 26(6):377–388. (Cited on page 51.)

Randić, M., Kleiner, A. F., and DeAlba, L. M. (1994). Distance/distance matrices. *Journal of Chemical Information and Computer Sciences*, 34(2):277–286. (Cited on page 25.)

Reid, K. B. and Beineke, L. W. (1978). Tournaments. In Beineke, L. W. and Wilson, R. J., editors, *Selected Topics in Graph Theory*, pages 169–204. Academic Press. (Cited on page 86.)

Sabidussi, G. (1966). The centrality index of a graph. *Psychometrika*, 31:581–603. (Cited on page 115.)

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409. (Cited on pages 45 and 61.)

Saul, L. K., Weinberger, K. Q., Ham, J. H., Sha, F., and Lee, D. D. (2005). Spectral methods for dimensionality reduction. In Schölkopf, B., Chapelle, O., and Zien, A., editors, *Semi-Supervised Learning*. MIT Press. (Cited on page 25.)

Sawant, A. P. and Bali, N. (2007). Diffarchviz: A tool to visualize correspondence between multiple representations of a software architecture. In *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 121–128. (Cited on page 95.)

Schoenberg, I. J. (1935). Remarks to Maurice Fréchet's article "Sur da définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 35(3):724–732. (Cited on page 14.)

Schönemann, P. H. (1970a). Fitting a simplex symmetrically. *Psychometrika*, 35(1):1–21. (Cited on page 15.)

Schönemann, P. H. (1970b). On metric multidimensional unfolding. *Psychometrika*, 35(3):349–366. (Cited on page 38.)

Schröter, A., Zimmermann, T., and Zeller, A. (2006). Predicting component failures at design time. In *Proceedings of the 6th International Symposium on Empirical Software Engineering*, pages 18–27. (Cited on page 94.)

Shepard, R. N. (1962). The analysis of proximities: multidimensional scaling with an unknown distance function, I, II. *Psychometrika*, 27:125–140, 219–246. (Cited on pages 13 and 43.)

Sibson, R. (1978). Studies in the robustness of multidimensional scaling: Procrustes statistics. *Journal of the Royal Statistical Society*, 40(2):234–238. (Cited on pages 18 and 62.)

Silva, J. G., Marques, J. S., and Lemos, J. M. (2005). Selecting landmark points for sparse manifold learning. In *Advances in Neural Information Processing Systems*. (Cited on page 27.)

Small, H. (1973). Cocitation in the scientific literature: A new measure of the relationship between two document. *Journal of the American Society for Information Science*, 24:265–269. (Cited on page 124.)

Small, H. (1999). Visualizing science by citation mapping. *Journal of the American Society for Information Science*, 50(9):799–813. (Cited on page 121.)

Storey, M.-A. D. and Müller, H. A. (1995). Manipulating and documenting software structures using shrimp views. In *Proceedings of the International Conference on Software Maintenance*, pages 275–284. (Cited on page 95.)

Storey, M.-A. D., Müller, H. A., and Wong, K. (1996). Manipulating and documenting software structures. In Eades, P. and Zhang, K., editors, *Software Visualization*. World Scientific. (Cited on page 93.)

Storey, M.-A. D., Wong, K., and Müller, H. A. (1997). Rigi: a visualization environment for reverse engineering. In *Proceedings of the 19th International Conference on Software Engineering*, pages 606–607. (Cited on page 93.)

Sugiyama, K. and Misue, K. (1995). A simple and unified method for drawing graphs: Magnetic-spring algorithm. In *Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94)*, pages 364–375. (Cited on pages 81, 85, and 89.)

Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125. (Cited on pages 80 and 94.)

Takane, Y., Young, F. W., and de Leeuw, J. (1977). Nonmetric individual differences multidimensional scaling: An alternating least-squares method with optimal scaling features. *Psychometrika*, 42:7–67. (Cited on page 42.)

ten Berge, J. M. F. (1991). A general solution for a class of weakly constrained linear regression problems. *Psychometrika*, 56(4):601–609. (Cited on page 52.)

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. (Cited on page 21.)

Torgerson, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, 17:401–419. (Cited on page 14.)

Torgerson, W. S. (1958). *Theory and methods of scaling*. Wiley. (Cited on page 14.)

Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–768. (Cited on page 49.)

Walshaw, C. (2000). A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing (GD '00)*, volume 1984 of *Lecture Notes in Computer Science*, pages 171–182. (Cited on pages 51 and 60.)

Wang, J. T.-L., Wang, X., Lin, K., Shasha, D., Shapiro, B. A., and Zhang, K. (1999). Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 307–311. (Cited on page 25.)

Ward, R. C. and Gray, L. C. (1978). Eigensystem compuation for skew-symmetric matrices and a class of symmetric matrices. *ACM Transactions on Mathematical Software*, 4(3):278–285. (Cited on page 79.)

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12. (Cited on page 22.)

White, H. D. and Griffith, B. C. (1981). Author cocitation: A literature measure of intellectual structure. *Journal of the American Society for Information Science*, 32:163–172. (Cited on page 124.)

White, H. D. and McCain, K. W. (1989). Bibliometrics. *Annual Review of Information Science and Technology*, 24:119–186. (Cited on page 121.)

White, H. D. and McCain, K. W. (1998). Visualizing a discipline: An author cocitation analysis of information science, 1972–1995. *Journal of the American Society for Information Science*, 49(4):327–355. (Cited on page 124.)

Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*. Oxford University Press. (Cited on page 18.)

Wills, G. J. (1997). NicheWorks – interactive visualization of very large graphs. In *Proceedings of the 5th International Symposium in Graph Drawing (GD'97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 403–414. (Cited on page 109.)

Yee, K.-P., Fisher, D., Dhamija, R., and Hearst, M. (2001). Animated exploration of dynamic graphs with radial layout. In *Proceedings of the InfoVis*, pages 43–50. (Cited on page 109.)

Young, F. W. and Hamer, R. M. (1987). *Multidimensional Scaling: History, Theory, and Applications*. Lawrence Erlbaum. (Cited on page 14.)

Young, G. and Householder, A. S. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22. (Cited on page 14.)

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473. (Cited on page 113.)

Zielman, B. and Heiser, W. J. (1996). Models for asymmetric proximities. *British Journal of Mathematical and Statistical Psychology*, 49:127–146. (Cited on page 74.)

Zimmermann, T. and Nagappan, N. (2008). Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th international conference on Software engineering*, pages 531–540. (Cited on page 94.)

Index

aesthetic criteria, 110
affiliation network, 129
algorithm
  Bellman-Ford, 22
  breadth-first search, 22
  Dijkstra, 22
  Floyd-Warshall, 22
  FM^3, 61
  GRIP, 61
  HDE, 61
  iterative, 42
  majorization, 47
all-pairs shortest-paths, 22
APSP, 22
asymmetricity, 74
asymmetry, 73
author, 123

barycenter, 34, 49
bibliographic coupling, 124, 126
bibliographic network, 124, 126
bibliographic proximity, 124, 125
bibliography, 121
  object, 122, 128
  operator, 122
  relationship, 122
bimension, 82, 86

centering, 29, 38, 39
centrality, 114
  a-priori, 97
  betweenness, 115
  closeness, 115
  hubs and authorities, 85
  PageRank, 94, 96, 104
  TrustRank, 97
  visual, 110
citation, 122, 123
classical scaling
  speedup, 25
clockwise orientation, 81
co-authorship, 124
co-citation, 124
collaboration, 124
collapsing, 49
column object, 38
component, 93
component level, 99
concentric rings, 109
connected component, 56, 125
constraint, 52
  distance, 109
  hard, 52
  radial, 109
  soft, 52
  strong, 52
  weak, 52
coupling graph, 125
crossing, 84, 110
cycles, 80
cyclic triple, 78, 88

data
  rectangular, 34

degree distribution, 67
dependency, 93, 101
dimension, 17, 23
  number, 18
dimensionality reduction, 21, 36
disparity, 43
dissimilarity, 14, 37
  squared, 39
distance
  average, 73
  erroneous, 16
  Euclidean, 14, 78
  group, 55
  infinite, 56
  interpolated, 99
  landmark, 34
  reconstruction of, 14
  shortest-path, 21, 23, 28, 39, 48, 61, 96, 128
  signed, 78
  squared, 42
double-centering, 15
dummy, 55, 114
Dynamic MDS, 55

edge
  coupling, 125
  curved, 84
  direction, 73
  length, 21
  non-reciprocated, 55
  number, 83
  reciprocated, 55
  significant, 125
eigengap, 20, 84
eigenpair, 17
eigenvalue, 16, 30, 66, 79
  complex, 75
  distribution, 25
  estimate, 19
  gap, 20
  imaginary, 75
  magnitude, 18, 37, 76
  sum, 83
eigenvector, 16, 30, 76, 79
  Laplacian, 49
energy, 51
Euclidean space, 13, 36
experiment, 58

feature, 36
focus, 113
force, 50
  attractive, 50, 85
  concentric, 81
  repulsive, 50
  rotating, 85

geometric rank, 97
graph
  bipartite, 36
  citation, 126
  complete, 21
  coupling, 125
  dependency, 94
  directed, 80
  disconnected, 55
  dynamic, 54
  evolving, 55
  meta-data, 94
  mixed, 55
  sparse, 22, 85
  strongly connected, 89
  tournament, 86
  underlying undirected, 55, 73
  undirected, 73
  weakly connected, 55
graph drawing, 21, 25, 45, 48
  clockwise, 86
  energy-based, 42, 57
  force-directed, 42, 48, 57, 81
  organic, 48

radial, 109
spectral, 21
group level, 99
grouping, 94, 123

hierarchy energy, 80
high-dimensional embedding, 38
hypothesis, 58

importance, 94
initial layout, 61, 66
initial solution, 48, 51, 54, 81
inner product, 15, 41
interpolation, 55, 84, 99
intrinsic dimensionality, 17, 59

Jaccard index, 125
Java, 100
JDK, 100

landmark, 27, 33
Landmark MDS, 33, 62
layout
dynamic, 55
initial, 21, 33
layered, 55
radial, 55
sequence, 54
linear combination, 53, 99, 110
local minimum, 51, 57, 58, 81
local neighborhood, 63

majorization, 45, 51, 53, 66, 112
convergence, 46, 51
inequality, 47
matrix
adjacency, 36, 81, 95, 123
centering, 15, 30, 34, 39
co-citation, 125
data, 36
degree, 95
distance, 22, 55, 89, 113
landmark, 34
Laplacian, 25, 49, 104
multiplication, 22
pivot, 27, 38
rectangular, 27
skew-symmetric, 73
skew-symmetric adjacency, 81
trace, 83
weight, 55, 111, 118
metro map layout, 117
missing values, 38, 41, 45

namespace, 100
neighborhood, 113
node
dummy, 55
focus, 113

objective function, 41
origin, 14, 29, 78, 81
overlap removal, 55

pattern analysis, 45
PCA, 36
penalty, 52, 53, 110
influence, 54
pivot, 27, 38, 39, 62
position, 31
Pivot MDS, 28, 38, 39, 62, 129
progressive, 33
polar coordinates, 84
polar transformation, 84, 88
power iteration, 18, 31, 61, 79, 85, 97
convergence, 18, 85
running time, 20
Principal Component Analysis, 36
probability distribution, 97
Procrustes roation, 62
Procrustes statistic, 62
projection, 39, 124
proximity, 93
asymmetric, 74

authors, 129
journals, 129
of nodes, 21
works, 128
psychometrics, 21, 43

radial constraints, 112
radial layout, 109
random walk, 97
repulsion, 50
residual error, 76
relative, 45
row object, 38

sampling
hybrid, 27
maxmin, 27
random, 27
scaling
absolute, 43
classical, 13, 41, 42, 48, 58, 61, 75, 95, 121, 128
distance, 41, 58
elastic, 45
metric, 43
nonmetric, 13, 43
skew-symmetric, 73
signed area, 77
similarity measure, 124
single-source shortest-paths, 22
singular value decomposition, 32, 37
skew-symmetry
edge number, 83
linear, 78
small world, 67
social network, 21, 55, 57, 113
social network analysis, 126
sociogram, 109
software system, 93
source code, 93
sparsity, 22, 81, 97
spectral decomposition, 16, 18, 75
spectrum, 18, 25
shift, 20
spline, 84
spring, 50
SSSP, 22, 33
statistics, 21
strain, 17, 23, 59, 76
contribution, 23
stress, 42, 59, 110
constraints, 53, 112
derivative, 45
distance, 111
majorization, 45
minimum, 45
sparse, 63
unweighted, 42
weighted, 43, 110
Sugiyama framework, 80, 94, 109
symmetry, 73, 93

target height difference, 80
target sociogram, 109
threshold, 47, 125
tournament, 86
traffic systems, 117
triangle, 78, 83
trivial solution, 45, 49, 53
Tube map, 117

unfolding, 37

Web of Science, 126
weight, 43, 110
constraints, 113
weighting scheme, 43, 110, 118
work, 45