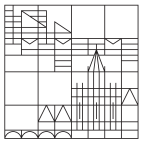
Drawing Dynamic Graphs by Stress Minimization

Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von
Martin Julius Mader

an der

Universität Konstanz



Mathematisch - Naturwissenschaftliche Sektion

Fachbereich Informatik und Informationswissenschaft

Tag der mündlichen Prüfung: 28. Mai 2014
1. Referent: Prof. Dr. Ulrik Brandes
2. Referent: Prof. Dr. Michael R. Berthold

Danksagung

Meinen Betreuern Ulrik und Michael für alles, das ich erlernen, erleben und erreichen durfte.
Meinen Kollegen und Freunden für eine gute Zeit an und außerhalb der Universität.
Meiner Familie für ihre unerschöpfliche Unterstützung und Liebe.

Danke!

Deutsche Zusammenfassung

Aufgrund des zunehmenden Interesses von Netzweranalysten an komplexen Netzwerken gibt es einen großen Bedarf an Netzwerkvisualisierungen, die visuelle Analyse und Exploration derselben ermöglichen. Dies gilt insbesondere für dynamische Netzwerke, die abstrakt gesehen aus einer Abfolge von Graphen bestehen. Die Schwierigkeit in der Visualisierung dynamischer Netzwerke besteht darin, eine kohärente Representation aufeinanderfolgender Graphen zu bestimmen, die den gängigen qualitativen Kriterien genügt, und gleichzeitig die mentale Struktur erhält, welche die betrachtende Person aus der vorhergehenden Netzwerkdarstellung aufgebaut hat.

Um das Kriterium dieser Stabilität zwischen individuellen Netzwerken zu erfüllen, müssen existierende Layout-Algorithmen für statische (also einzelne) Graphen entsprechend modifiziert werden. In dieser Arbeit wird *Stress-Minimierung*, ein gängiges und erwiesenermaßen leistungsstarkes Layoutverfahren, erweitert, um verschiedene Strategien zum Zeichnen dynamischer Graphen zu realisieren: *Aggregation* erzeugt ein globales Layout für alle individuellen Graphen der Sequenz. *Verankerung* schränkt die Wegbewegung eines Knotens von dessen Position in einem Referenzlayout, zum Beispiel dem Vorhergenden, ein, während *Verkettung* diese Einschränkung simultan zwischen allen Instanzen eines Knotens realisiert. Es werden dabei technische, und – anhand von exemplarischen Illustrationen – auch qualitative Aspekte beleuchtet.

Diese fundamentalen Strategien werden dann systematisch bezüglich ihrer Fähigkeit untersucht, einen guten Kompromiss zwischen genauer und lesbarer Darstellung einzelner Graphen und dem Erhalt der mentalen Struktur zwischen Graphen zu erlangen. Bisher wurde die Effektivität dieser Abwägung beinahe ausschließlich durch qualitative Illustrationen demonstriert, oder aber in Nutzerstudien, welche die kognitive Perzeption solcher Darstellungen betrachten, untersucht. Hier wird die bisher große Lücke zwischen diesen beiden Evaluationsansätzen geschlossen, in dem die Durchführung der Algorithmen als technisches Experiment angesehen wird, so dass eine quantitative Bewertung von Hypothesen über das Verhalten der Algorithmen möglich wird. Um die Experimente durchführen zu können, wird ein Generator für dynamische Graphen entwickelt, der auf der Basis von realen Daten zufällige, aber dennoch strukturierte Sequenzen erzeugt. Darüberhinaus wird ein Messapparat vorgestellt, mit dem sich die Ausgaben der Algorithmen quantifizieren lassen. Insgesamt erlaubt es die Implementation der verschiedenen Strategien durch Stress-Minimierung somit, durch Verwendung traditioneller experimenteller Methodologie nachzuweisen, inwieweit die realisierten Strategien den Erwartungen entsprechen, und inwieweit der Kompromiss zwischen individueller Layoutgüte und dynamischer Stabilität gesteuert werden kann.

Contents

1. Introduction

Does individual behavior like alcohol consumption affect how friendships emerge for young adolescents? How can an institution with high employee turnover and forced job rotation preserve accumulated knowledge? And are there systematic patterns that affect how countries form international trade agreements? Although these questions are on completely different levels, they have in common that evolving relations need to be considered to answer them – in these cases, friendships, advice seeking and trade relations.

This kind of data is commonly modeled as a sequence of *networks* comprising the set of entities under investigation and the relations among them, including the semantic meaning of entities, relations, and possibly, their various attributes. Of course, evolving, or *dynamic*, networks are ubiquitous in many areas of research and practice, like computer science, biochemistry, or, as just demonstrated, social science, which is the focal application area considered here.

Social network analysts use sophisticated statistical models to research the interplay of evolving behavioral attributes and network relations (see Knecht (2008); Lazega, Lemercier, and Mounier (2006); Manger, Pickup, and Snijders (2012) for the examples above). With these, hypotheses are evaluated based on fit of model parameters and statistical significance. However, it is undeniable that *network visualization* is equally important for network research, because it is key to not only communicate network data, but also to explore it (Freeman, 2000), and thus facilitates building hypotheses, gaining insight about characteristic structures, or detecting anomalies (Klovdahl, 1981).

This thesis is about techniques to automatically obtain visualizations of a dynamic network in a form that social scientists (and, in fact, researchers of many other fields) are familiar with. Corresponding to the input sequence of networks, we are to obtain a sequence of so called *node-link diagrams*, where network entities are displayed as points (or, rather, shapes defined by a single point), and relations as straight lines, as illustrated in Figure 1.1.

The characterizing trade-off in the dynamic situation is between the individual quality of each diagram and the persistence of features over the sequence (Brandes and Wagner, 1997a). In other words, each diagram should be a good representation of the corresponding cross-sectional network, and at the same time, a *mental map* of the structure should be preserved as much as possible to relate the individual frames with less cognitive effort (Eades, Lai, Misue, and Sugiyama, 1991).

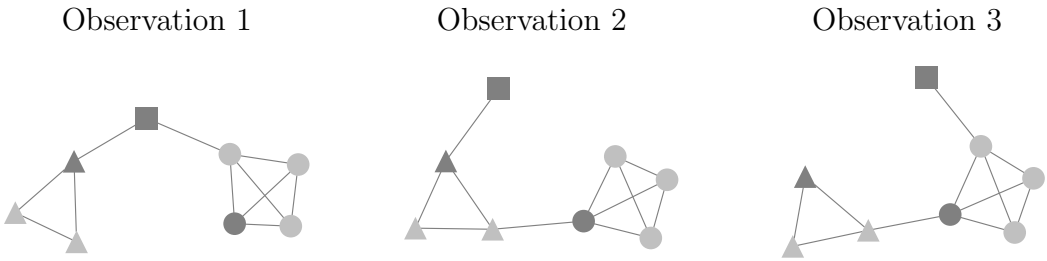


Figure 1.1.: Visualization of a sample dynamic network comprising 3 observations by node-link diagrams.

Computing a *graph layout* is at the core of automatic network visualization. Graphs are the mathematical abstraction of networks, that allow algorithmic processing, and a layout defines the geometric representation of a graph's elements in the plane. We here extend *stress minimization*, the state-of-the-art algorithm for computing a graph layout in the single-graph or *static* case (Brandes and Pich, 2009), to the dynamic scenario. To incorporate preservation of the mental map, we adopt three general approaches that aim to stabilize node positions throughout the sequence.

Do these methods work well? The most prominent ways to evaluate dynamic graph layout algorithms have been by either qualitative illustrations on select examples, or studies investigating user performance in different settings. Taking an *algorithm engineering* perspective, we here try to fill the gap, and quantitatively provide insight about the method's behavior by purely algorithmic experimentation.

The thesis is structured into three main chapters:

Chapter 2 – Graphs and graph drawing Static graph layout algorithms are the fundament of any dynamic extensions. After providing basic terminology and defining the layout problem for our scenario of node-link representations for general input graphs, we survey the two most prominent approaches to solve this problem. *Force-directed* methods are very widely used in practice. Here, the graph is likened to a physical system of attracting and repelling forces, and an equilibrium solution is sought to obtain a layout.

Recently, *stress minimization*, a well-studied method originating in the field of psychometrics, was rediscovered for the use in graph drawing. The objective is to optimally represent graph-theoretic distances of the input graph in the layout. The method was shown to be more robust, efficient and effective than force-directed methods, and yet retains the flexibility needed for adaption to specific problems like dynamic graph drawing. Since stress minimization is the basis of the dynamic adaptions presented in this thesis, it is reviewed in more detail.

Chapter 3 – Dynamic graph drawing methods In the dynamic scenario, the temporal dimension constitutes an additional level of complexity. As for the static case, each layout should faithfully reveal individual structure, but additionally, the sequence of layouts must be stable so as to not destroy a viewer's mental map. The predominant principle to achieve this stability is to restrict node movement between observations.

Previous work on dynamic layout algorithms using this principle can be categorized into three approaches: Maximum stability is achieved in *aggregation* approaches where fixed node positions are obtained from the layout of an aggregate of all graphs in the sequence. Alternatives that allow for explicit control over node stability are based on *anchoring* nodes to reference positions (usually the node's position in the previous layout), or *linking* nodes to instances of themselves that are close in the sequence.

We formulate these approaches as variants of stress-minimization via aggregation of input distances for the aggregation approach, and via consistent, weakly constrained stress functions to realize anchoring and linking. Properties of variant approaches are qualitatively discussed by means of a small artificial and a larger real-world example.

Parts of this chapter have been published in Brandes, Indlekofer, and Mader (2012) and Brandes and Mader (2012).

Chapter 4 – Experimental evaluation The qualitative demonstrations of Chapter 3 point us to interesting phenomena, but do these generalize well beyond those select examples? For algorithms like stress minimization, which are hard to characterize analytically, experimentation is needed to test assumptions against reality.

We employ an experimental design following algorithm engineering principles: The goal is to support or falsify hypotheses about how different model parameters affect outcomes of variant approaches, and thus provide quantitative evidence for differential behavior. To be able to test our hypotheses, we need to address two problems: First, there is a lack of sufficiently large benchmark data for dynamic networks; how can random, yet structured and application-typical, data be generated? Second, how can the relevant properties of layout quality and stability be measured for graphs of different sizes and structure? Both these problems are dealt with in a novel way.

Although we discover finer subtleties, we find support for hypotheses regarding the trade-off between individual layout quality and stability. Furthermore, the linking approach turns out to be preferable to other variants.

Parts of this chapter have been published in Brandes and Mader (2012).

We summarize the main results and contributions of this thesis in Chapter 5 and outline promising endeavors for future work.

2. Graphs and graph drawing

> *"The simplest, most efficient construction [of a network drawing] is one which presents the fewest meaningless intersections, while preserving the groupings, oppositions, or potential orders [among entities]."*
>
> *(Bertin, 1983, page 271)*

Visualization of networks is a challenging topic because its purpose is not just mere presentation of relational data. A good visualization must reveal the structural and semantic information of the network in a most precise way, to facilitate exploration or communication of this information.

Network visualization can draw on two major streams of research, information visualization of networks (Herman, Melançon, and Marshall, 2000) and graph drawing (Di Battista, Eades, Tamassia, and Tollis, 1999; Kaufmann and Wagner, 2001). The emphasis in information visualization is on visualization design, navigation, and interactivity. In contrast, properties and construction of geometric representations of the abstract structural model underlying a network – the *graph* – are more central to graph drawing, and the focal aspect in this work.

A plethora of methods has been developed to obtain a drawing for an individual graph which is given completely in advance – the so called *static* case. Often, these methods are tailored to certain classes of input and designed to meet both general and application-dependent criteria and conventions. The main motivation behind this thesis is visualization of social networks. In this area, input graphs are very diverse, and can hardly be put into a single class of networks, which calls for a general-purpose graph drawing method. Additionally, social scientists are very accustomed to a representation of networks in which entities of the network are depicted as simple shapes (usually points) and relations as simple curves between entities (usually straight lines).

The predominant approaches to obtain drawings of general graphs adhering to these standards are *force-directed* methods, that solve the problem by relaxing a physical model of attractive and repulsive forces imposed on the input graph. However, in recent years, *stress minimization*, an instance of *multi-dimensional scaling* (MDS) techniques long established in statistics, (re-)gained popularity as a method for general-purpose graph drawing. This method is reasonably simple to implement and has been shown to be more robust, effective and efficient than force-directed methods. Yet, it retains the flexibility inherent to force-directed methods that is essential for adaptation towards specific criteria, and specifically enables our extensions for drawing *dynamic* graphs.

Besides providing basic terminology needed throughout this thesis, this chapter gives a more detailed problem description of general-purpose static graph drawing. Afterwards force-directed methods are shortly reviewed, because these are the basis of most approaches for dynamic graph drawing so far. A large part is then devoted to stress minimization and its application in graph drawing, as it is the foundation of our dynamic approaches.

2.1. Notions and problem description

As network visualization is a topic addressed in very different domains, there often is quite some confusion about terminology. It is therefore necessary to define notions and the visualization problem treated in this thesis in detail.

First, the terms *network* and *graph* must be distinguished. The term network encompasses the conceptual entirety of relational data: the entities involved, the relational structure between entities, attributes associated with entities or relations, and especially, the semantic meaning of entities, relations, and attributes. For example, the actors in a social network might be either humans, institutions or organizational units (or a mixture thereof), and the relations might be affiliation, friendship or enmity.

A graph is the mathematical abstraction of relational data, which allows the data to be handled analytically and algorithmically. The entities are modeled as a set of vertices, and the relations as a set of edges between those. In its simplest form, the graph represents the structure of relations, however, additional attributes of either entities or relations may be encoded as weight functions defined on the sets of vertices and edges. Central graph concepts and definitions needed in this work are given next.

Graph definitions A graph is denoted by $G = (V, E)$, where $V = V(G)$ is a set of n *vertices* and $E = E(G)$ is a set of m *edges*. For convenience, vertices are indexed with natural numbers, i.e., $V = \{1, \ldots, n\}$.

A graph is called *directed* if $E \subseteq V \times V$, that is, an edge is an ordered pair of vertices v and w that is denoted by (v, w). The graph is called *undirected* if edges are unordered pairs, denoted by $\{v, w\}$. The remaining definitions will be given with respect to undirected graphs; however all are naturally transferable to directed graphs. A *loop* is an edge $e = \{v, v\}$. If the edge set of a graph is a multi-set the graph is called a *multi-graph*, and an edge occurring more than once in E is called a *multi-edge* or *parallel edge*. A graph is *simple* if it does not contain any loops or multi-edges. Two vertices v and w are *adjacent* if there is an edge $e = \{v, w\} \in E$, and, in this case, v and w are *incident* to e and edge e incident to v and w. Also, two edges e_1 and e_2 are incident if they share a common vertex, i.e., $e_1 = \{u, v\}$ and $e_2 = \{v, w\}$. An arbitrary pair of vertices, regardless of whether connected by an edge or not, is called a *dyad*.

A *path* between two vertices v and w is a sequence of incident edges $\{v = u_0, u_1\}$, $\{u_1, u_2\}$, …, $\{u_{k-1}, u_k = w\}$, where $u_0, \ldots, u_k \in V$. The *length* of the path is $k \in \mathbb{N}$, i.e., the number of edges it contains. If there is any path between two vertices, they are called *connected*, and if there is none they are called *disconnected*. Furthermore, a graph is *connected* if any pair of vertices is connected, and *disconnected* otherwise. If a graph is disconnected, its connected subgraphs are called (connected) *components*.

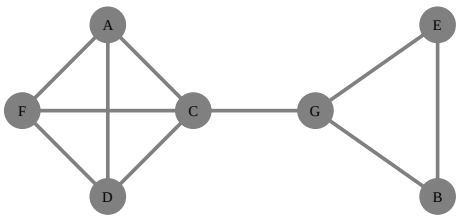
A path with length k between two vertices is a *shortest path* if there is no other path connecting these vertices with length $k' < k$. The length k of a shortest path between vertices v and w defines their *graph-theoretical* or *shortest-path distance*, denoted by $d(v, w) = d_{vw} = k$. The shortest-path distance of a vertex to itself is defined as $d(v, v) = 0$, and two disconnected vertices v and w are defined to be at distance $d(v, w) = \infty$.

Graph Layout Any network visualization should aim to effectively reveal the information contained in the network, where structural information encoded in the underlying graph naturally is most important. The central task therefore is to choose an appropriate form of representation and to prescribe a mapping of a graph's elements to features in the visual space. In the most general sense, graph *layout* is that part of the mapping that is concerned with topological and geometric properties.

Depending on the form of representation, determining a layout resolves to different algorithmic problems. The most common graphical representation for networks are *node-link diagrams*, in which vertices are depicted as points (or, more precisely, graphical elements described by a single position), and edges are depicted as lines linking their endpoints, see Figure 2.1a. The layout problem then is to determine positions for the vertices, and potentially positions for bend or control points for edges. The quality of the layout is central for the quality of the whole diagrams. This is because positional differences are the most accurately perceived graphical attributes (Cleveland and McGill, 1984). If the layout is of low quality, even the best graphical design (in terms of using other graphical attributes such as shape, color, size, etc.) or interaction mechanisms can only attenuate the problems of poor legibility and interpretation artifacts.

A different form of representation for graphs is the *matrix* representation (Bertin, 1983), a table indexed by vertices in rows and columns, containing boolean values corresponding to adjacency, see Figure 2.1b. The layout problem here is to find a permutation of vertices such that structural information is most accessible, usually in the form of dense blocks around the diagonal. Again, a good permutation is crucial to obtain meaningful and legible representations.

It should be noted that both forms accommodate arbitrary graph input. This is important for visualizing social networks, because these do not constitute a formally boundable class of graphs, even though they exhibit some general tendencies such as sparseness and local clustering. While graph structure is represented completely in plain versions of both representations, the other attributes of a network can be incorporated by varying



(a) node-link diagram

	A	D	F	C	G	B	E
A	0	1	1	1	0	0	0
D	1	0	1	1	0	0	0
F	1	1	0	1	0	0	0
C	1	1	1	0	1	0	0
G	0	0	0	1	0	1	1
B	0	0	0	0	1	0	1
E	0	0	0	0	1	1	0

(b) matrix representation

Figure 2.1.: Different forms of graph representations.

graphical attributes like shape, color or size. Clearly, these choices are more dependent on the data and context, and in general easier to implement.

We here restrict our scope to node-link diagrams. They are more intuitively accessible than matrix representations, and very familiar to many researchers. Indeed, their utility for social network analysis was already demonstrated by Moreno (1953), There, node-link diagrams are referred to as *sociograms*.

We further restrict the layout problem to *straight-line* representation of edges, where lines representing edges are completely specified by the positions of their incident vertices. That is, they do not have any additional bend-points (*polyline* representation), and are not restricted to run along certain directions (e.g., only horizontal or vertical lines as in *orthogonal* representations).

In summary, the visualization problem addressed in this thesis is determining a layout for a straight-line representation of general graphs:

Given: A simple, undirected graph $G = (V, E)$.

Find: A layout $P = (p_1, \ldots, p_n)$ of two-dimensional positions $p_i \in \mathbb{R}^2$ for every vertex $i \in V$, such that structural information is represented in a most effective way.

Note, that, in general, we will not make the distinction between the terms actors, nodes, vertices, and points, and between the terms ties, links, edges, and lines.

Aesthetic criteria It remains to discuss the factors that constitute an effective layout. As the general goal is to facilitate exploration and communication, effectiveness is not necessarily to be compared to artistic beauty. Rather, a layout is effective if it is unambiguous, presents the structural information faithfully, and is readable, that is, can be understood in a short time.

Certainly, design principles for general data presentation, known from graphic design (Bertin, 1983; Tufte, 1983) and Gestalt theory (Koffka, 1935), are important for, and apply to, graph layout. While incorporating these principles is already demanding for a human designing a layout manually, it is clear it is even harder to do so for algorithms for automatic layout generation.

However, there are some commonly acknowledged objectives that can be injected to graph layout algorithms, and by which those can be evaluated for effectiveness, called *aesthetic criteria*. These criteria are at the foundation of the graph drawing discipline – both standard textbooks about graph drawing by Di Battista, Eades, Tamassia, and Tollis (1999) and Kaufmann and Wagner (2001) treat them in the very beginning. Surprisingly, however, little work is available on evaluating them from the perspective of human cognition (Purchase, Cohen, and James, 1997; Ware, Purchase, Colpoys, and McGill, 2002) and from a technical perspective (Taylor and Rodgers, 2005).

Bennett, Ryall, Spalteholz, and Gooch (2007) provide an extensive compilation of various aesthetic criteria, and point to work about validation and evaluation. Those criteria considered most relevant for our case of application-independent straight-line representations are the following:

- *Vertices should have distinct positions.* Naturally, vertices having the same position cause ambiguity.
- *Edges should be of more or less the same length.* As length is a prominent visual variable, different lengths may impose different visual emphasis. Additionally, uniform edge length induces that structural groupings are visible in the layout.
- *Vertices should be distributed well over the drawing area.* If vertices are spread out, white space is reduced, and the drawing area is utilized well. This facilitates recognizing details in the vicinity of vertices and reduces clutter.
- *The number of meaningless edge crossings should be kept small.* Edge crossings make it difficult to follow paths in the graph, especially if edges cross at a small angle. This criterion is maybe the one which is most agreed upon, and also the one most evaluated empirically.
- *Symmetries in graph structure should be visible in geometric symmetries.* Symmetries are easily perceived, and an important structural characteristic in many applications.

For specific applications and purposes, there may be many more criteria to observe. For most of them, optimization is computationally intractable even in isolation, at least for general graphs. Since, in addition, the various criteria are frequently contradictory, general-purpose layout algorithms are usually heuristic in nature.

2.2. Force-directed methods

The most popular approach to general-purpose graph drawing are *force-directed* methods, colloquially known as *spring embedders*. We shortly review these methods because of their relation to the layout method of our choice that is described in the next section, and because they are the basis of previous approaches for dynamic graph layout, see Chapter 3. For more details, see Brandes (2001) and Di Battista, Eades, Tamassia, and Tollis (1999).

The name "force-directed" is due to the fundamental concept behind those methods: a graph is likened to a physical system of attractive and repulsive forces, and a solution of the layout problem is obtained at a force equilibrium of the system. The vertices are modeled as objects repelling each other and edges are modeled as springs of a given length, binding adjacent vertices together. Intuitively, the repulsive forces care for a good distribution of vertices in the area, whereas modeling the edges as springs with uniform ideal length addresses the criterion of uniform edge length.[1]

Although related concepts were already in use in the context of VLSI design (Quinn and Breuer, 1979), the seminal work in the field of graph drawing is a short note of Eades (1984). It proposes logarithmic strength springs to model edges instead of the natural linear springs as in Hooke's law, i.e., the spring force is proportional to the logarithm of its deviation from the ideal length. Non-adjacent vertices repel each other following Coulomb's law for electrically charged particles, i.e., the repelling force is proportional to the inverse square of their distance.

Starting from some initial layout the system is relaxed by iteratively moving vertices according to their net force vector, which is the sum of all attractive and repulsive forces acting on a vertex. In each iteration forces are calculated synchronously for all vertices; thus, the cost for force calculation per iteration is linear in the number of edges for attractive forces and quadratic in the number of vertices for repulsive forces. Afterwards, the position update is also performed synchronously for all vertices, where each vertex is moved by a (constant) fraction of its net force vector to prevent excessive movement. Eventually, the system reaches a stable state of a local force equilibrium, and the resulting positions are adopted for the straight-line drawing.

[1] Note that it is $\mathcal{NP}$-hard to decide whether a general graph has a drawing with uniform edge length in any number of dimensions (Johnson, 1982).

Benefits and drawbacks Generally, force-directed methods yield satisfactory results for small to medium-sized graphs, that is, graphs with up to a few hundred vertices. The layouts obtained usually exhibit uniform node distribution and edge lengths. Also, even without explicitly modeling the layout criterion for symmetries, these are most often reflected in the resulting layout, and congruent structures in a graph typically have congruent geometric representations (Brandenburg, Himsolt, and Rohrer, 1996).

However, the main beauty of the force-directed model is that it is easy to grasp because of its physical analogy, that we quite easily understand from our everyday experience in the real world. Furthermore, this intuitive access makes the method very flexible and extensible, as specific purposes can be addressed by using additional or alternative forces. For example, Huang and Eades (1998) provide an extension for clustered graphs. They introduce virtual vertices representing each cluster and virtual edges connecting the vertices of a cluster to their representative, where additional forces control the cohesion within clusters and separation between clusters. Other examples are Sugiyama and Misue (1995), who use forces of an external magnetic field to yield drawings of directed graphs such that edges point in roughly the same direction, or Brandes and Wagner (1998) who introduce additional forces on control points of Bézier curves for layouts with curved edges. Indeed, modifying the input graph structure and adding or modifying forces in the physical model is also key to the extensions for dynamic graph layout, as shown in Chapter 3.

Despite these beneficial properties that led to the great popularity of force-directed methods, there are also severe drawbacks. The main problem is convergence to a stable state, which can be very slow, and even worse, is not guaranteed at all. The system could get stuck in, for example, an oscillating or rotating state, or only reach a stable state due to global dampening of the movement from net force vectors (Frick et al., 1995). Hence, the stable state is not necessarily at a local minimum of the intrinsic energy of the physical system. Some of these issues are addressed in the many modifications and improvements of the classical spring embedder of Eades that are shortly described in the following.

Another problem is that force-directed methods require configuring many parameters, from the actual physical model itself to constants involved in force calculations to controlling the dampening of the net force vector. Of course, this problem is further aggravated when additional mechanisms are employed to improve convergence.

Lastly, all force-directed methods are susceptible to the initial layout from which layout computation starts. Earlier publications most often use a random initial placement and claim that initialization does not effect the result obtained, but indeed affects the efficiency in terms of the number of iterations needed to reach a stable state (Kamada and Kawai, 1988; Frick et al., 1995). However, this is mainly because the graphs used for validation at that time were very small. With increasing graph sizes it is now known that "bad" initialization does also have a strong influence on the quality of the result.

Modifications and improvements Due to the popularity of force-directed methods, there are a myriad of modifications tailoring them to certain applications, like the examples mentioned above. Here, we will only very briefly review variants that introduced fundamental developments addressing some of the shortcomings of spring embedders.

One of the most popular, and often only available layout algorithm implemented in common software tools for network visualization, is the variant of Fruchterman and Reingold (1991). Besides proposing a variation of attractive and repulsive forces to allow for more efficient calculation, they increase robustness by introducing the concept of *temperature*. That is, the fraction of movement due to the net-force vector is successively decreasing with the number of iterations performed already. Second, they speed up calculation of repulsive forces using a grid technique, such that these are only evaluated for vertices in the vicinity of the vertex at hand.

The variant of Frick, Ludwig, and Mehldau (1995) further extends the concept of temperature by determining a local temperature at each vertex that individually slows movement of vertices that are found to oscillate or being part of a rotating subgraph. Additionally, they add gravitational forces to keep disconnected components from drifting apart. Finally, the iteration mechanic is changed such that in each iteration only one vertex is processed and immediately updated.

Davidson and Harel (1996) adopt a different perspective on relaxing the physical system to reach a stable state. Instead of calculating force vectors and moving vertices along the force, they formulate the system in terms of an *energy function*, and a layout is computed by minimizing the energy of the system. Still, the constituent components of the energy function conform to the attractive and repulsive forces of previously mentioned force-directed approaches. However, additional components are proposed to incorporate the number of edge crossings or distance between nodes and (non-incident) edges explicitly in the energy function. The energy of the system is here minimized by an optimization procedure called *simulated annealing* that originated in statistical mechanics, but is also applicable to complex combinatorial problems (Kirkpatrick, Gelatt, and Vecchi, 1983). Although this approach is maybe the most flexible and is robust against local minima, it is hard to parametrize appropriately and, more importantly, very slow; therefore, its use is prohibitive for larger graphs.

A big step towards applicability of force-directed methods to large graphs with several thousand vertices came in the early years of 2000 with the emergence of *multi-scale* or *multi-level* approaches (Hadany and Harel, 2001; Walshaw, 2001; Harel and Koren, 2002; Gajer, Goodrich, and Kobourov, 2004; Hachul and Jünger, 2004). The fundamental principle is to abstract or *coarsen* the graph multiple times, while retaining the most important features. Then layout calculation is done in a top-down fashion, beginning with the coarsest version, and reintroducing finer detail successively in each level. The referenced approaches mainly differ in the way a graph is coarsened – different schemes of edge contractions and vertex filtrations are proposed – and the force-directed method chosen for layout within a level.

Although it was actually one of the first variants of the original spring embedder, the method of Kamada and Kawai (1988) is presented at the end of this section, due to its strong relation to the layout method of our choice. It differs significantly from other force-directed methods, because the physical model is not based on the combination of attractive force that try to bring adjacent vertices close together and and repulsive forces that try to generally keep vertices apart from each other. Instead, each pair of vertices is connected by a spring with an ideal length proportional to the corresponding shortest-path distance. Similar to Davidson and Harel (1996) the physical system is relaxed by minimizing an energy function, rather than moving vertices along force vectors. Kamada and Kawai stated this objective function E in terms of squared differences between the desirable shortest-path distances and the actual distances in the layout,

$$E = \sum_{i<j} \frac{K}{d(i,j)^2} \left(L \cdot d(ij) - \|p_i - p_j\|\right)^2 , \tag{2.1}$$

where $\sum_{i<j}$ is a shorthand notation for $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n}$, i.e., summation over all unordered pairs of vertices , K is a constant and L is the desirable length of a single edge. The intuition is that the layout is the more pleasing the better shortest-path distances are represented; thus, the objective is to find a layout with minimal energy. Algorithmically, a local minimum is obtained using a modified two-dimensional Newton-Raphson method, which iteratively moves a single vertex according to the partial derivates of the energy function. This optimization procedure is, again, quite inefficient, and the originally proposed method therefore not applicable to larger graphs unless used within a multi-level approach. However, the resulting layout for small graphs compare favorably with other force-directed methods (Brandenburg, Himsolt, and Rohrer, 1996), and therefore, the method is also found very often in software tools for graph visualization.

2.3. Multidimensional scaling and stress minimization

Interestingly, the objective function that Kamada and Kawai (1988) used in their spring embedder was already extensively studied a decade earlier in a more general context. It is known as *stress* (Kruskal and Wish, 1978), and is studied in the field of *multidimensional scaling* (MDS), a family of dimension-reduction techniques that originated in statistics, and particularly, in psychometrics. Given measured dissimilarity between a set of objects, the general goal of MDS is to find positions for these objects in a low-dimensional space, such that dissimilarities are best represented. Here, we only report on basic concepts important for application in graph drawing. These are extensively treated in Pich (2009) and Klimenta (2012). Further discussions of history, theory and application of MDS in the original statistical context are found in textbooks like Cox and Cox (2001) or Borg and Groenen (2005).

It is almost ironic that MDS is one of the earliest computer-implemented methods for drawing social networks, as presented in Kruskal and Seery (1980), and seemingly was already applied in the late 60's. Their work is mentioned in the often cited annotated bibliography for graph drawing algorithms (Di Battista et al., 1994). However, MDS approaches and, particularly, stress minimization, did not receive much attention in the graph drawing community until Gansner, Koren, and North (2004) reported on a then long-established, robust and efficient optimization strategy for stress minimization, called majorization, and thus re-popularized the method for graph drawing.

It turned out that stress minimization in its current form compares favorably to force-directed methods, including multi-level approaches, both in terms of quality and efficiency (Brandes and Pich, 2009). At the same time it allows similar flexibility and is reasonably easy to implement. In the following, stress minimization and its application to static graph drawing are described in detail.

Stress minimization Let $O = \{1, \ldots, n\}$ be an a set of n objects, and let

$$D = (\delta_{ij})_{i,j \in O} \in \mathbb{R}^{n \times n}$$

be a matrix of object *dissimilarities*. Usually, the dissimilarity matrix is required to contain nonnegative elements ($\delta_{ij} \geq 0$), to be symmetric ($\delta_{ij} = \delta_{ji}$), and to have a zero diagonal ($\delta_{ii} = 0$). The general purpose of multidimensional scaling is to determine d-dimensional positions $p_i \in \mathbb{R}^d$ for every object $i \in O$ such that the Euclidean distances in the d-dimensional space resemble the given dissimilarities as closely as possible, that is,

$$\delta_{ij} \approx \|p_i - p_j\| \ ,$$

where $\| \cdot \|$ denotes the Euclidean norm. For any given configuration $P = (p_1, \ldots, p_n)$ this objective is quantified using a parameterized *stress function* $\mathsf{stress}(P)$,

$$\mathsf{stress}(P) = \sum_{i<j} \omega_{ij} \left(\delta_{ij} - \|p_i - p_j\|\right)^2 \ , \tag{2.2}$$

where $W = (\omega_{ij})_{i,j \in O}$ is a weight matrix whose entries determine the contribution of each pair $i, j \in O$. Since stress is defined as the weighted sum of squared dissimilarity-representation errors, the objective is to find a configuration of minimum stress.

The application of stress minimization to graph drawing is straight-forward: The objects under consideration is the set of vertices V, and dissimilarities between vertices are chosen to reflect the layout criteria at hand. A two-dimensional configuration P with minimum stress then immediately transforms into a layout for a straight-line drawing of the graph.

Dissimilarities The similarity of stress to the energy function that Kamada and Kawai (1988) used in their spring embedder immediately suggests that shortest path distances are a plausible choice for vertex dissimilarities. An important aspect of shortest path distances d of a graph $G = (V, E)$ is that, when d is regarded as a function

$$d : V \times V \to \mathbb{R}$$

the pair (V, d) forms a *metric space*, since the following holds for any $i, j, k \in V$:

$$\begin{array}{rcll} d(i,i) & = & 0 & \text{(reflexivity)} \\ d(i,j) & = & d(j,i) & \text{(symmetry)} \\ d(i,k) & \leq & d(i,j) + d(j,k) & \text{(triangle inequality)} \end{array}$$

The metric properties of shortest path distances facilitate their direct use as dissimilarities for stress minimization and other MDS techniques. Setting $\delta_{ij} := d(i, j)$, the stress term of each dyad corresponds to the squared error of representing a shortest path as a straight line with unit length edges. This leads to several intuitive consequences concerning layout criteria in a low-stress layout: Since a pair of vertices connected by an edge have the minimal shortest path distance of 1, adjacent vertices should be placed close to each other. Additionally, larger distances will force non-adjacent vertices to be spread out over the drawing area. As the error of distance representation contributes by its square in stress, and thus, larger errors are penalized more, edges can be expected to display roughly the same length. Finally, symmetric structures in the graph are likely to exhibit similar representations in the layout, since they have an identical backbone of shortest paths.

Computing shortest paths in a graph is a well-studied algorithmic problem. If the task is to compute shortest paths from a single vertex to all others – called *single-source shortest-path problem* (SSSP) – a simple *breadth-first search* (BFS) yields shortest path distances in $\mathcal{O}(m)$ time, if the input graph has a uniform weight associated with each edge. This unweighted case is most typical for application in graph drawing. However, there might be reason to consider weighted shortest paths, where the weight of a path is the sum of weights of its edges. Solving the problem in this case requires more sophisticated algorithms. The most prominent one is Dijkstra's algorithm (Dijkstra, 1959), which enhances a BFS-type graph traversal with relaxation of intermediate shortest path information to solve the SSSP for graphs with nonnegative edge weights, and can be implemented to run in $\mathcal{O}(n \log n + m)$ time by using dedicated data structures (Fredman and Tarjan, 1987). To solve the *all-pairs shortest-path problem* (APSP), i.e., determine shortest path distance between all pairs of vertices, algorithms for SSSP can be executed starting from each node. However, for weighted graphs the Floyd-Warshall algorithm (Floyd, 1962; Warshall, 1962) that solves APSP directly in $\mathcal{O}(n^3)$ time is often more efficient in practice due to small constant factors. It is a dynamic program that successively incorporates more and more intermediate vertices to be considered for shortest paths. For an extensive treatment of graph traversal and shortest path algorithms, see Cormen, Leiserson, Rivest, and Stein (2001).

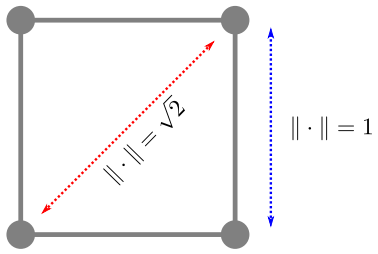


Figure 2.2.: Realizing shortest path distances is not generally possible even for trivial graphs such as a four-cycle: The two length 2 distances (red) cannot both be realized while exactly representing all four length 1 distances (blue) at the same time.

Cohen (1997) proposes a different metric distance to be used for stress minimization, which he calls *linear-network distance*. It is based on a physical analogy: The graph is likened to a resistive electrical circuit, where each edge represents a conductor with conductance according to its weight (or uniform conductance for unweighted graphs). The distance between to vertices is then defined as the total resistance between their representatives in the circuit, and thus, as opposed to considering only the shortest path, considers multiple paths connecting them. With more and stronger paths, linear network distance gets smaller, resulting in a tighter representation of clustered structures, whereas shortest path distances are supposedly "more appropriate for regular structures".

In the following, we only consider shortest path distances as dissimilarities for stress minimization, as is most common. However, linear network distance nicely demonstrates, that specific layout criteria might be addressed and emphasized by engineering input dissimilarities used in such methods.

Weights Stress minimization yields an exact representation of dissimilarities only if these actually are Euclidean distances from a Euclidean space having the desired dimension. It is easy to see that an exact representation generally is not possible for shortest path distances, even for trivial graphs, for example, a simple four-cycle, see Figure 2.2. In the natural layout that depicts the four-cycle as a square with unit edge length, all shortest paths of length 1 are represented perfectly, but the two shortest paths of length 2 between opposite vertices have Euclidean distance $\sqrt{2}$. If trying to better represent one of the length 2 shortest paths, while keeping distance 1 between adjacent vertices, the other length 2 shortest path necessarily gets smaller.

Weights W can be used to influence representation accuracy for certain dyads, and were originally introduced to mend robustness of stress minimization against outliers of measured input dissimilarities. Usually, weights are not set individually, but are given as a function of the input dissimilarities, where the form $\omega_{ij} = \delta_{ij}^q,\ q \in \mathbb{R}$ is most common

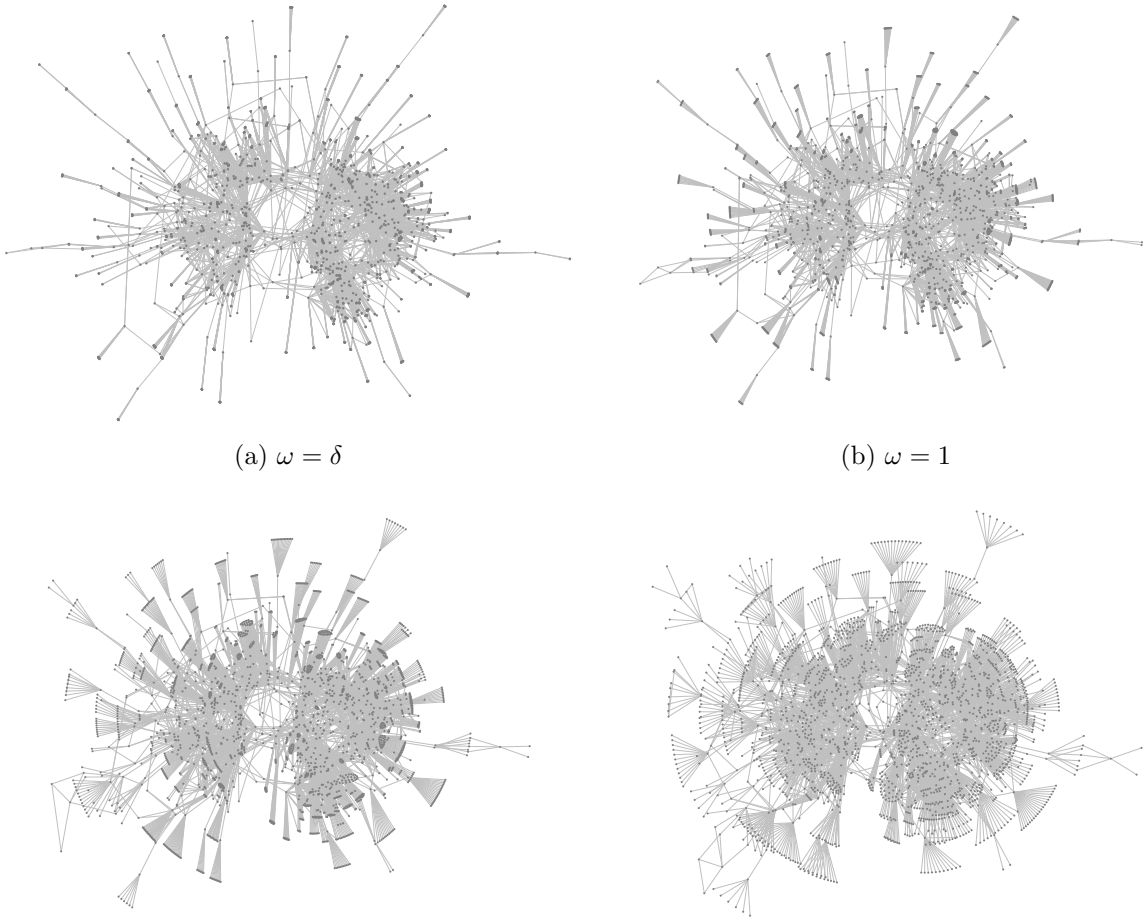
(indices will be tacitly omitted if the context is clear). Positive powers enforce a more accurate representation of large dissimilarities, while negative ones put more emphasis on representation of small dissimilarities. Cohen (1997) considers $q \in \{0, -1, -2\}$ for the purpose of graph drawing, and calls the respective stress functions *absolute*, *semi-proportional*, and *proportional* stress.

Stress functions with these weighting schemes were earlier discussed in the context of MDS. All dissimilarities are treated equally in the original proposal of stress (Kruskal, 1964a), i.e., $\omega = \delta^0 = 1$. The stress function then conforms to the sum of squared absolute fit residuals. The *nonlinear mapping* algorithm of Sammon (1969), which is often referred to as "Sammons's mapping", turns out to be an instance of MDS (Kruskal, 1971), as it effectively uses stress with weights $\omega = \delta^{-1}$ as the objective function. Lastly, McGee (1966) introduced *elastic scaling* and named the corresponding objective function *work*, which is stress with weights $\omega = \delta^{-2}$. In this case, dissimilarities are not fitted by their absolute residuals, but rather by relative errors, since

$$\delta_{ij}^{-2} \left(\delta_{ij} - \|p_i - p_j\|\right)^2 = \left(1 - \frac{\|p_i - p_j\|}{\delta_{ij}}\right)^2 ,$$

and thus, deviation of Euclidean distance and input dissimilarity only contributes as a fraction of the latter.

For the application in graph drawing, weights are crucial to control the general appearance of layouts. This is demonstrated in Figure 2.3 that shows layouts obtained by stress minimization with the mentioned weighting schemes. The depicted graph `esslingen1` is a social network comprising actors and political events in the German town of Esslingen in the 19th century (Lipp, 2000). Using absolute stress results in a layout that emphasizes larger distances (Figure 2.3b), due to the naturally larger variance of absolute residuals in that region of the input domain. Although this leads to a reasonable representation of global structure, local details like the many peripheral tree-like structures, or locally denser regions in the center of the network, are hard to detect. Inherently, this effect is even stronger if a weighting scheme with a positive power of dissimilarities is employed, as shown in Figure 2.3a for $\omega = \delta$. The more larger dissimilarities are down-weighted, the more local detail is recognizable in the resulting layouts. Semi-proportional stress (Figure 2.3c) reveals more local detail in the periphery of the example network, but denser regions in the center still appear very cluttered. If the application at hand does not specifically require faithful representation of large dissimilarities, proportional stress (Figure 2.3d) seems to be the best compromise to display global structure and local detail at the same time, and also to conform well with the aesthetic criteria of uniform vertex distribution and uniform edge length. This is not too surprising, as proportional stress exactly matches the established objective function in the spring embedder of Kamada and Kawai (1988). Hence, when stress is used in the context of graph drawing, we will assume $\omega = \delta^{-2}$ unless otherwise noted.



(a) $\omega = \delta$ (b) $\omega = 1$

(c) $\omega = \delta^{-1}$ (d) $\omega = \delta^{-2}$

Figure 2.3.: Layouts of the main component of the `esslingen1` social network ($n = 2075,\ m = 4769$) obtained by stress minimization using different weighting schemes $\omega = \delta^q$ with $q \in \{1, 0, -1, -2\}$.

2.3.1. Optimization procedure

Since no closed form is known to minimize stress as given by Equation 2.2 directly, iterative heuristic methods have to be used to solve this optimization problem. While stress-related problems where first solved by *gradient-descent* methods (Kruskal, 1964b), i.e., iteratively moving objects according to partial derivates of the stress function, it turned out that an approach called *majorization* (de Leeuw, 1977) is by far more robust and efficient, and provably converges. The idea is to not minimize the stress function directly, but instead to minimize a function that is greater or equal than the stress function - hence the name majorization - and easier to solve algebraically and computationally. In the following, we shortly review the majorization process; for further details the reader is referred to de Leeuw (1977, 1988) and Gansner, Koren, and North (2004).

By expanding the terms of Equation 2.2 the stress function can be written as

$$\mathsf{stress}(P) = \sum_{i<j} \omega_{ij}\delta_{ij}^2 + \sum_{i<j} \omega_{ij}\|p_i - p_j\|^2 - 2\sum_{i<j} \omega_{ij}\delta_{ij}\|p_i - p_j\| \quad , \tag{2.3}$$

and can be rewritten in matrix form as

$$\mathsf{stress}(P) = \sum_{i<j} \omega_{ij}\delta_{ij}^2 + \mathrm{tr}(P^T L P) - 2\ \mathrm{tr}\left(P^T B(P) P\right) \quad , \tag{2.4}$$

where $\mathrm{tr}(M)$ is the trace, i.e., the sum of all elements of the diagonal of a matrix M, M^T denotes the transpose of matrix M, and matrices L and $B(P)$ are defined as

$$L_{i,j} = \begin{cases} \sum_{k\neq j} \omega_{ik} & \text{if } i = j \\ -\omega_{ij} & \text{if } i \neq j \end{cases}$$

and

$$B(P)_{i,j} = \begin{cases} -\dfrac{\omega_{ij}\delta_{ij}}{\|p_i - p_j\|} & \text{if } i \neq j \text{ and } \|p_i - p_j\| \neq 0 \\ 0 & \text{if } i \neq j \text{ and } \|p_i - p_j\| = 0 \\ -\sum_{k\neq i} B(P)_{i,k} & \text{if } i = j \end{cases} .$$

A function that majorizes stress is

$$T(P, Q) = \sum_{i<j} \omega_{ij}\delta_{ij}^2 + \mathrm{tr}(P^T L P) - 2\ \mathrm{tr}\left(P^T B(Q) Q\right) \quad , \tag{2.5}$$

for any configuration Q having the same dimensions as P. Clearly, $\mathsf{stress}(P) = T(P, P)$. Furthermore it can be shown by use of the Cauchy-Schwarz inequality that

$$\mathrm{tr}\left(P^T B(P) P\right) \geq \mathrm{tr}\left(P^T B(Q) Q\right) \quad ,$$

and hence,

$$\mathsf{stress}(P) \leq T(P, Q) \quad , \tag{2.6}$$

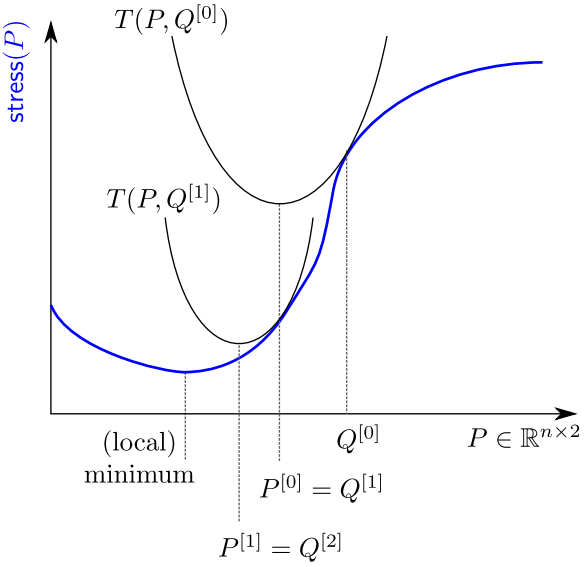


Figure 2.4.: Illustration of the majorization process (reproduced from de Leeuw 1988).

thus fulfilling the criteria of a majorant. Now, function $T(P, Q)$ is a quadratic form and has only global minima – with translation being the only degree of freedom – that can be found by differentiating $T(P, Q)$ by P and thus solving

$$LP - B(Q)Q = 0. \tag{2.7}$$

Putting all this together gives rise to an iterative optimization process: Starting from a layout $Q^{[0]}$ we want to obtain a layout $P^{[0]}$ with $\mathsf{stress}(P^{[0]}) < \mathsf{stress}(Q^{[0]})$, which we can now compute by assigning $P^{[0]}$ as the minimizer of $T(P, Q^{[0]})$, since then

$$\begin{array}{ccccccc} \mathsf{stress}(P^{[0]}) & \leq & T(P^{[0]}, Q^{[0]}) & < & T(Q^{[0]}, Q^{[0]}) & = & \mathsf{stress}(Q^{[0]}) \\ & & | & & | & & | \\ & & \text{Eq. 2.6} & & P^{[0]} = \arg\min_P T(P, Q^{[0]}) & & \text{Eqs. 2.4, 2.5} \end{array} \tag{2.8}$$

assuming $Q^{[0]}$ is not already the minimizer of $T(P, Q^{[0]})$. In the next iteration, we take $Q^{[t]} = P^{[t-1]}$, and repeat until either the relative change of layouts or relative change in stress is below a certain threshold ϵ, i.e.,

$$\frac{\|P^{[t-1]} - P^{[t]}\|}{\|P^{[t-1]}\|} < \epsilon \quad \text{or} \quad \frac{\mathsf{stress}(P^{[t-1]}) - \mathsf{stress}(P^{[t]})}{\mathsf{stress}(P^{[t-1]})} < \epsilon \quad , \tag{2.9}$$

or for a fixed number of iterations. Figure 2.4 illustrates the iterative majorization process. The advantage of majorization over earlier gradient methods is that the sequence of stress values is always non-increasing, i.e., $\mathsf{stress}(P^{[0]}) \geq \mathsf{stress}(P^{[1]}) \geq \cdots \geq \mathsf{stress}(P^{[t]})$ and always converges to a stationary point of the stress function, which usually is a local minimum (de Leeuw, 1988).

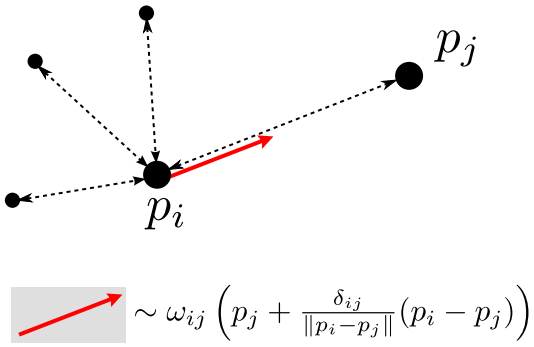


Figure 2.5.: Vote of vertex j for the new position of vertex i in the localized method for stress minimization (solid arrow), based on weight ω_{ij}, desired distance δ_{ij} and current Euclidean distance $\|p_i - p_j\|$ (dashed arrow).

Localized majorization For one step of the majorization cycle, we need to compute the minimizer of the majorant $T(P, Q)$, which can be done by solving Equation 2.7 via matrix inversion or by use of equation solvers. However, Gansner, Koren, and North (2004) provide an elegant alternative way to obtain a sequence of layouts with non-increasing stress by majorization in a localized manner, very much in line with the optimization procedure used for spring embedders. In each cycle, we iterate over all vertices of the graph and regard all positions except for the current vertex i as fixed. Every other vertex j *votes* for a desired placement of vertex i, based on the current positions $p_i^{[t]}$ and $p_j^{[t]}$, the desired distance δ_{ij}, and the weight ω_{ij} set for this dyad, see Figure 2.5. Then, all votes are combined to the new position $p_i^{[t+1]}$ by taking the weighted average over all votes, yielding the update function

$$p_i^{[t+1]} = \frac{\sum_{j \neq i} \omega_{ij} \left(p_j^{[t]} + s_{ij}^{[t]} \, (p_i^{[t]} - p_j^{[t]}) \right)}{\sum_{j \neq i} \omega_{ij}} \quad , \tag{2.10}$$

where

$$s_{ij}^{[t]} = \begin{cases} \dfrac{\delta_{ij}}{\|p_i^{[t]} - p_j^{[t]}\|} & \text{if } \|p_i^{[t]} - p_j^{[t]}\| > 0 \\ 0 & \text{otherwise} \end{cases} \quad .$$

Using this update function, it is guaranteed that stress of the sequence of layouts obtained over the iterations is monotonically reduced, and thus, that the procedure converges to a local minimum of the stress function. Algorithm 1 summarizes the localized stress minimization method for graph drawing. Obviously, the algorithm is reasonably easy to implement, needing only simple data structures and operations.

Algorithm 1: Stress minimization for graph drawing

Input: Undirected, connected graph $G = (V, E)$,
initial positions $P = (p_i),\ i \in V$,
dissimilarity matrix $D = (\delta_{ij}),\ i, j \in V$ (usually shortest-path distances),
threshold ϵ

Output: Positions P at a local minimum of $\mathsf{stress}(P)$ wrt. D

$W \leftarrow (\omega_{ij}) = (\delta_{ij}^{-2})$

repeat

  foreach $i \in V$ **do**

$$p_i \leftarrow \frac{\sum_{j \neq i} \omega_{ij} \left(p_j + s_{ij} \ (p_i - p_j)\right)}{\sum_{j \neq i} \omega_{ij}}, \text{ with } s_{ij} = \begin{cases} \dfrac{\delta_{ij}}{\|p_i - p_j\|} & \text{if } \|p_i - p_j\| > 0 \\ 0 & \text{otherwise} \end{cases}$$

until P *or* $\mathsf{stress}(P)$ *change by less than* ϵ *(cf. Eq. 2.9)*

2.3.2. Finding good local minima

Stress minimization appears to share the drawback of other force-directed methods that iterative stress reduction in general only yields a local minimum which may be far from an optimal layout. Which local minimum is reached heavily depends on the initial layout given to the optimization algorithm, therefore the quality of the solution depends on the quality of the initial layout.

A naive and computationally costly approach to obtain a good solution is employing a so called *multi-start* strategy, that is, to perform stress minimization from multiple random initializations and to choose the solution with lowest stress (Kruskal, 1964b). A different viable method is to start with a solution in a higher output dimension, and successively use the higher-dimensional result as initialization for computation in a lower dimension, by dropping dimensions with least variance. The rationale behind this dimension reduction approach is that there likely are much less different local minima in higher-dimensional solutions than in lower-dimensional ones (Groenen and Heiser, 1996). A more involved, and not always successful, strategy is *tunneling* (Groenen and Heiser, 1996), where after a local minimum is found, a different configuration having the same stress is sought by minimizing a tunneling function, and optimization repeated from this configuration. Another approach is *distance smoothing* (Groenen et al., 1999), that aims to mend irregularities in the stress function by introducing a parameterized smoothing function to the distance calculation.

Alternatively, stress minimization can be initialized with a more systematically constructed configuration, instead of random ones. The most popular candidate is the first practical method available for MDS, called *classical scaling* (CMDS). It was independently developed by Torgerson (1952) and Gower (1966), and therefore is also known as *Torgerson-Gower scaling*. Classical scaling is a purely analytic approach that results in

an essentially unique solution. Although sole reliance on a CMDS solution for initialization is discouraged in general MDS applications (Buja and Swayne, 2002), it turns out that CMDS solutions serve as very suitable initial layouts for application in graph drawing, as experimentally verified in Brandes and Pich (2009). Indeed, the latter work establishes that in the context of graph drawing the two-step process of initializing stress minimization with a CMDS solution surpasses previous force-directed approaches. We will go into more detail after shortly reviewing the basics of CMDS next.

Classical scaling Let, as for stress minimization, $D = (\delta_{ij})_{i,j \in V} \in \mathbb{R}^{n \times n}$ denote the matrix of input dissimilarities, however assuming for now that those are actually Euclidean distances from a d-dimensional Euclidean space. The objective of CMDS is to obtain a layout $P = (p_1, \ldots, p_n)^T \in \mathbb{R}^{n \times d}$ such that $\delta_{ij} = \|p_i - p_j\|$. Instead of fitting input dissimilarities directly, as in stress minimization, CMDS takes a detour by utilizing that coordinates P can be reconstructed from the matrix of their *inner products* B,

$$B = PP^T, \; b_{ij} = p_i^T p_j \; ,$$

and that B in turn can be obtained from dissimilarities without knowledge of P. The latter can be seen from squaring the objective function, that is,

$$\delta_{ij}^2 = \|p_i - p_j\|^2 = (p_i - p_j)^T (p_i - p_j) = p_i^T p_i - 2 p_i^T p_j + p_j^T p_j \; ,$$

which, solved for $p_i^T p_j$ yields

$$p_i^T p_j = -\frac{1}{2} \left(\delta_{ij}^2 - p_i^T p_i - p_j^T p_j \right) \; .$$

Furthermore, we may assume that P is centered in the origin of the coordinate system, that is, $\sum_{i=1}^n p_i = 0$, since distances do not change under translation. Then, the term $-p_i^T p_i - p_j^T p_j$ can be expressed by averages over the rows, over the columns, and over all entries of the matrix of squared dissimilarities, yielding a coordinate-independent representation of inner products

$$b_{ij} = -\frac{1}{2} \left(\delta_{ij}^2 - \frac{1}{n} \sum_{k=1}^n \delta_{ik}^2 - \frac{1}{n} \sum_{l=1}^n \delta_{lj}^2 + \frac{1}{n^2} \sum_{k=1}^n \sum_{l=j}^n \delta_{kl}^2 \right) \; . \tag{2.11}$$

It should be noted that the above operation on the squared dissimilarities within the braces can be performed for the whole matrix at once, and is known as *double centering*. The desired configuration P can now be recovered from matrix B in the following way: Let

$$B = U \Lambda U^T$$

be the *eigendecomposition* of B, where Λ is the diagonal matrix of eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ of B, and U is the orthonormal matrix of its eigenvectors $u_1, \ldots, u_n$. As

D is assumed to be a matrix of d-dimensional Euclidean distances, B has real-valued nonnegative eigenvalues, and the decomposition can be rewritten as

$$B = U\Lambda U^T = \left(U\Lambda^{1/2}\right)\left(U\Lambda^{1/2}\right)^T =: PP^T \ .$$

By the same assumption we know that B has rank d, and hence, has exactly d positive and $n-d$ zero eigenvalues. Therefore, the final d-dimensional configuration is obtained by safely ignoring the dimensions corresponding to zero eigenvalues, and setting

$$P = U_d\Lambda_d^{1/2} \ ,$$

where Λ_d is the diagonal matrix of the first d eigenvalues, and U_d is the matrix of associated eigenvectors.

Now, input dissimilarities cannot be assumed to be exactly Euclidean generally; especially, in the graph drawing context, shortest-path distances are known to be non-Euclidean but for the most simple graphs, as demonstrated earlier. The practical solution in this case is to use the d largest positive eigenvalues and the corresponding eigenvectors to reconstruct the configuration, as those contribute most towards faithful representation (Gower, 1966), resulting in

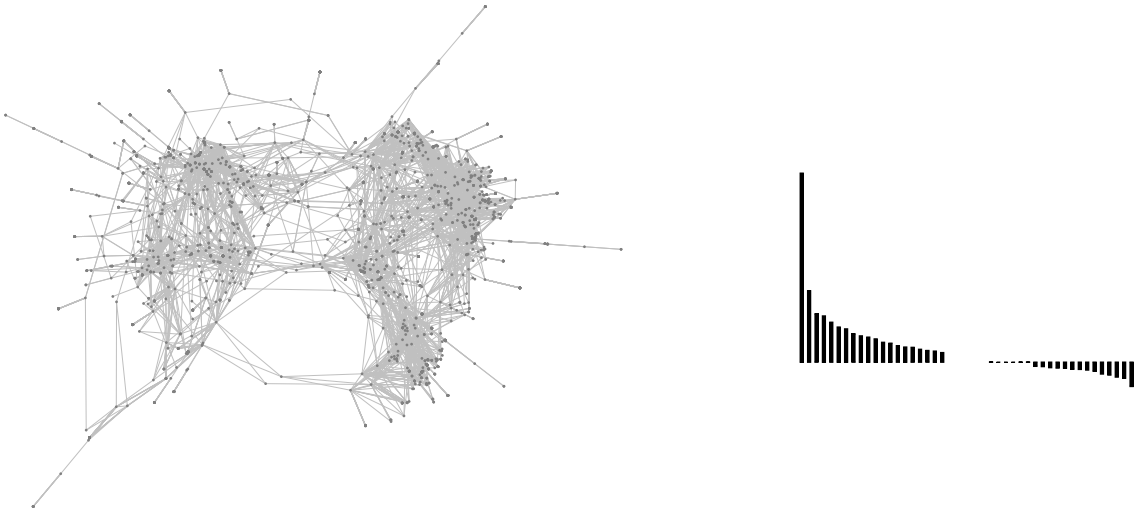
$$P = U_d\left(\Lambda_d^+\right)^{1/2}, \ \lambda_i^+ = \max\{\lambda_i, 0\} \ . \tag{2.12}$$

Consequently, dimensions corresponding to small positive eigenvalues, which contribute only marginally, as well as those corresponding to negative eigenvalues, which indicate "non-Euclideaness" of input dissimilarities, are ignored. It is therefore worth to have a look at the spectrum, i.e., the distribution of eigenvalues, to assess the so called *intrinsic dimensionality* of the data, and thus, to assess the amount of information loss due to scaling and the expected quality of representation. Indeed, if d is not given in advance the spectral distribution may be used to determine an appropriate dimensionality (Mardia, 1978; Sibson, 1978). Since $d := 2$ in our context, it is desirable that the two largest eigenvalues are reasonably large in magnitude compared to the others, whereas large negative eigenvalues are undesirable, as those signify that MDS may not be appropriate. Fortunately, these requirements are sufficiently met for most graphs, although certain graph classes, like low-diameter[2] graphs, can be problematic (Brandes and Pich, 2009).

Figure 2.6 shows the layout obtained by classical scaling with $d = 2$ for the `esslingen1` network, together with the extremal regions of the spectrum, i.e., the distribution of eigenvalues, of B.

We can now confirm the earlier statements, that CMDS is a purely analytic technique involving no heuristics, and indeed, yields a unique solution up to rotation and reflection. Moreover, Mardia (1978) proves that a CMDS solution as given by Equation 2.12 also is

[2]The diameter of a graph is defined as the maximum length of all its shortest paths



(a) CMDS solution (b) extremal regions of $\Lambda(B)$

Figure 2.6.: Layout of the `esslingen1` network obtained by classical scaling, and extremal regions of the spectrum of the inner product matrix B.

optimal with respect to fitting inner products derived from dissimilarities (Equation 2.11) to actual inner products of the solution P, since it minimizes the loss function *strain*,

$$\mathsf{strain}(P, D) = \left\| B - PP^T \right\|^2 \ .$$

However, optimality w.r.t. strain causes a CMDS solution to be inappropriate for most graph drawing applications, similar to unweighted stress: Because strain is a least-squares sum of inner product fit, CMDS tends to overemphasize large distances over small ones, and thus fails to reveal local structures (see Figures 2.3 and 2.6). On the other hand, global structure is usually well represented.

2.3.3. Effectiveness and efficiency

Apart from personal experience and general consensus, effectiveness of stress minimization is extensively studied and experimentally verified by Brandes and Pich (2009). Employing a hypotheses-based experimental design using a test suite of various graphs, they conclude that a two-step approach, consisting of CMDS followed by stress majorization of proportional stress, yields much better results than other force-directed methods, including multi-level approaches (Gajer, Goodrich, and Kobourov, 2004; Hachul and Jünger, 2004). Specifically, they show that, when faithful representation of shortest path distances is imposed as a criterion for layout quality, the two-step approach generally yields better goodness-of-fit in terms of the distribution of output distances w.r.t. actual shortest path distances. Second, they provide evidence that initialization by CMDS indeed routinely leads to low-stress layouts after few iterations of stress minimization.

Scalability to larger graphs is not a prime focus in this thesis, since with standard straight-line representations, visual complexity drastically increases with increasing graph size in most cases. However, many approaches dealing with visual complexity of large graphs still require a full layout, that is, vertex positions, to begin with (e.g., Gansner et al. 2011; Zinsmaier et al. 2012). Similar approaches are conceivable for dynamic graphs as well.

Stress-minimization seems to be impracticable for larger graphs, as the individual components effectively have a computational complexity similar to that of standard force-directed approaches:

- *Distance computation* – For the general case of unweighted graphs, the full shortest path distance matrix is most efficiently obtained by performing n breadth-first searches taking $\mathcal{O}(nm)$ time.
- *CMDS* – Computation of the inner product matrix B trivially takes $\mathcal{O}(n^2)$ time. However, the more intensive task is to obtain the spectral decomposition of B. As only the two largest eigenvalues and corresponding eigenvectors are needed, *power iteration* is a reasonably simple and efficient way to obtain them, see Golub and Van Loan (1996) for details. Essentially, a random non-zero vector repeatedly is multiplied by B until convergence, resulting in the eigenvector u_1 corresponding to the eigenvalue λ_1 with largest absolute value.[3] The eigenvector u_2 may be obtained by orthogonalization against the first one. Each iteration thus takes $\mathcal{O}(n^2)$ time, where the convergence rate is determined by the *eigengap* $|\lambda_1|/|\lambda_2|$. For most practical instances the number of iterations can be regarded a constant factor.
- *Stress majorization* – The local majorization process, as shown in Algorithm 1, clearly needs $\mathcal{O}(n^2)$ time per iteration. Again, the number of iterations can usually be considered a constant due to reasonable initialization by CMDS.

Fortunately, there are ways to deal with the complexity for the latter two steps, that also imply savings on the distance computations.

Classical scaling can be approximated very well with a method called *PivotMDS* (Brandes and Pich, 2007). The main idea is to only use a pivot set $\mathcal{U} \subset V$ of $k \ll n$ vertices and their distances $D_{\mathcal{U}} \in \mathbb{R}^{n \times k}$ to all other vertices, reducing the complexity of distance computation to $\mathcal{O}(km)$ time. It is shown that the right singular vectors ν_1, ν_2 of the inner product matrix C – which is obtained analogous to Equation 2.11 from $D_{\mathcal{U}}$ – are estimates for the eigenvectors u_1, u_2 of B.[4] Again, ν_1, ν_2 can be calculated by power iteration. Assuming the size of the pivot set k is constant, PivotMDS can be implemented to run in linear-time to obtain an approximation of the CMDS solution.

[3] Note that the first eigenvalues (ordered by absolute value) are usually positive in our context.

[4] The originally proposed algorithm for PivotMDS introduces artifacts with respect to aspect ratio. This can be corrected by changing the last statement of Algorithm 1 in Brandes and Pich (2007) to $x \leftarrow C\left(1/\sqrt[4]{v_1^T C^T C v_1}\right) v_1$, and analogous for y (Klimenta, 2012).

In much the same manner, the complexity of the stress minimization step can be reduced by simply omitting the contribution of certain dyads. This is often referred to as *sparse stress*. Ignoring dyads is easily accomplished due to the weighting in stress, as it corresponds to setting the weight of a dyad to 0 which, in the original statistical context, was beneficial to deal with missing values. Practically, zero-weight dyads can be omitted in the sum of Algorithm 1, effectively reducing its complexity. For graph drawing, it is natural to ignore dyads that have a large shortest path distance, since by $\omega_{ij} = \delta_{ij}^{-2}$ their contribution is small relative to dyads connected by short shortest paths. There is not much substantial work on sparsification schemes, however, Pich (2009, Chapter 6) shows by qualitative examples, that a naive approach often yields good results: For each node only a *local* neighborhood is considered, say, the l nearest nodes. All neighborhoods and corresponding distances can be calculated by performing n truncated breadth-first searches, yielding a $\mathcal{O}(ln)$ complexity for distance calculation, which then is also the complexity of an iteration in stress minimization. The results can be further improved qualitatively by also considering the distances already calculated for PivotMDS.

A more recent approach based on sparse stress is suggested by Gansner, Hu, and North (2013). They propose to minimize a sparse stress function that only considers the actual edges in the graph, and augment the function with an entropy term caring for vertex distribution reminiscent of repulsive forces in spring embedders.

2.3.4. Flexibility

The last property of stress minimization that has to be discussed, and indeed is essential for its application to dynamic graph drawing, is its flexibility. Although it is not as intuitively accessible as the force-directed paradigm, stress minimization offers almost the same expressive power. This is because there are several degrees of freedom that allow for sophisticated layout modeling. We may alter the given graph's structure resp. stress terms, admissible positions, dissimilarities, and weights to incorporate constraints.

Constraints can be separated into two classes, depending on the degree of which algorithms need to satisfy them:

- *Hard constraints:* The solution has to strictly satisfy the constraints. This induces a *feasible* region of solutions, from that one with minimal stress is sought. One way to introduce hard constraints into stress minimization is to project an intermediate solution back to the feasible region after each iteration. A problem is that some effort is needed to prove that the projected configuration is closest to the intermediate unconstrained solution, and to prove that convergence properties still hold.
- *Weak constraints:* In a weakly constrained solution constraints do not necessarily need to be strictly enforced, but a larger violation of constraints should lead to higher stress. This is typically achieved by augmenting the stress function with a

penalty term $c(P)$ representing deviation from constraints in the configuration P. The resulting constrained stress function $\mathsf{stress}^C(P)$ is usually formulated as a convex combination of regular stress and the penalty term (Borg and Lingoes, 1980), i.e.,

$$\mathsf{stress}^C(P) = (1-\alpha)\ \mathsf{stress}(P) + \alpha\ c(P)\ , \tag{2.13}$$

where parameter α controls the influence of constraints. Often, the penalty term $c(P)$ can be expressed by deviations of desired and Euclidean distances between vertices by augmenting the input graph with auxiliary virtual vertices; then the standard majorization approach can be used for minimization, and optimality and convergence properties still hold.

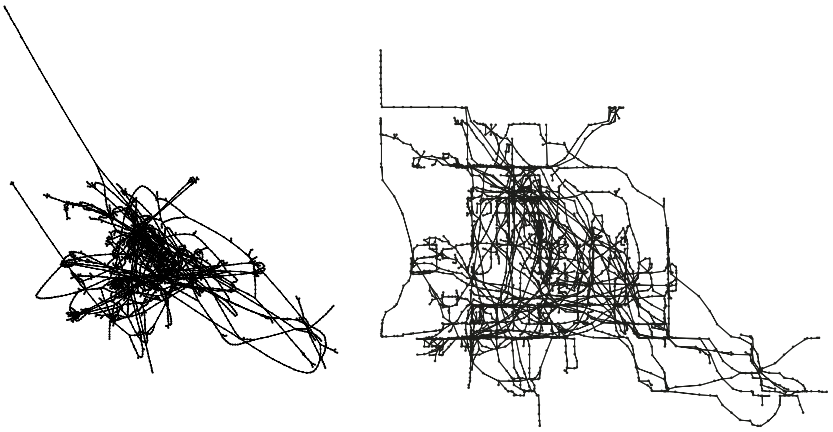
Several approaches showcasing the flexibility of stress minimization are shown in Figure 2.7. Projections to feasible regions are used, e.g., in Dwyer, Koren, and Marriott (2006c), to strictly respect a given *orthogonal ordering*, that is, the order of objects along axes, see Figure 2.7a. This framework has been generalized in Dwyer, Koren, and Marriott (2006b) to arbitrary linear separation constraints that enforce given horizontal or vertical separation of dyads. The same authors extended this further by augmenting the stress function with weak constraints considering length of certain paths to improve a given layout while preserving its topology (Dwyer, Marriott, and Wybrow, 2009).

Brandes and Pich (2011) propose to use stress with weak constraints to obtain a *centrality drawing* of graphs, where vertices are to be aligned to radial orbits according to a given centrality index, see Figure 2.7b. To enforce the desired radii $r_i, i \in V$, a dummy vertex o representing the origin is introduced, and the penalty term is formulated as

$$c(P) = \sum_{i \in V} r_i^{-2} \left(r_i - \|p_i - o\|\right)^2\ .$$

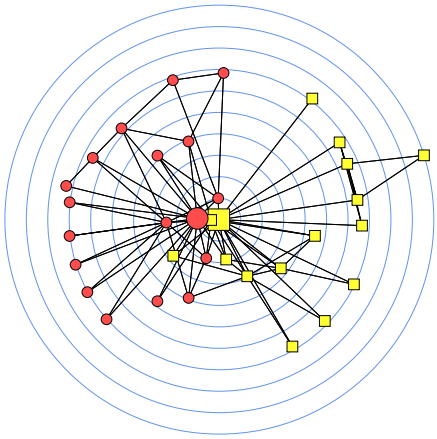
Note that, due to the structure of the penalty term, virtually no modification is needed for the majorization algorithm. Following Borg and Lingoes (1980), parameter α from Equation 2.13 is gradually increased from 0 to 1 during minimization, such that focus shifts from faithful structural representation to realization of desired radial distances. Therefore, structural features are retained in the final layout, which is not necessarily the case if only the penalty term is minimized directly.

The following two examples are not modifications of stress minimization targeting hard or weak constraints, but rather work by engineering the input graph, stress terms and used dissimilarities. The *binary stress* model of Koren and Çivril (2009) is a combination of a stress term for adjacent vertices with desired zero distance, and a stress term for all dyads with desired unit distance. Originally designed as a sparse model for efficient computation, it characteristically produces space-filling layouts with a circular shape, because of the latter stress term, see Figure 2.7c. Gansner and Hu (2009) propose a modified stress model for removing overlap between rectangular-shaped vertices after a regular layout considering vertices as points has been calculated, see Figure 2.7d. Here, input dissimilarities are Euclidean distances of the initial layout which are enlarged by

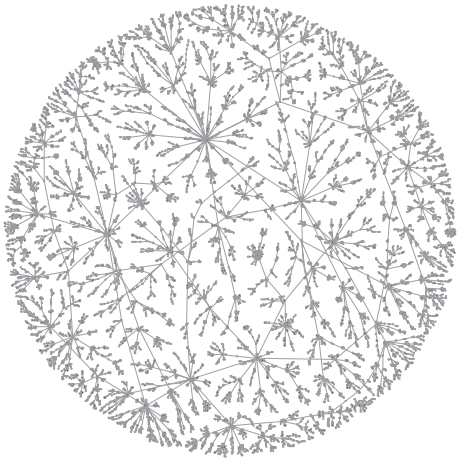


geographic positions layout preserving orthogonal ordering

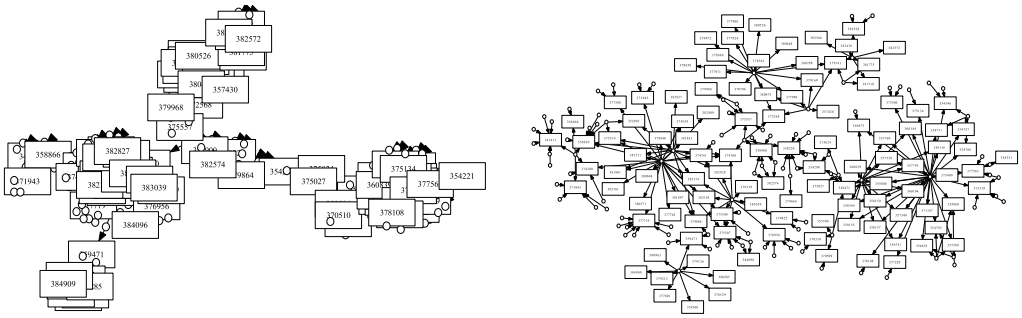
(a) Layout of an Internet backbone network preserving orthogonal order induced by geographic positions (Dwyer et al., 2006c).



(b) Centrality drawing of a social network (Brandes and Pich, 2011).



(c) Space filling drawing of a large tree (Koren and Çivril, 2009).



initial layout after overlap removal

(d) Overlap removal (Gansner and Hu, 2009).

Figure 2.7.: Different variants of stress minimization for sophisticated layout modeling.

a factor corresponding to the amount of overlap between two vertices. Additionally, this stress model is only optimized over edges in a *Delaunay triangulation* of the input graph which serves two purposes: Triangulation helps to preserve relative positions of the initial layout due to rigidity, and also improves computational cost, since the number of considered dyads is linear in the number of vertices.

3. Dynamic graph drawing methods

> *"If we are going to get an intuitive and theoretical grasp of the underlying processes in dynamic networks, we need network visualizations that capture dynamics and generative mechanisms. Dynamic network visualization of these processes enables researchers to explore this untapped terrain, [and] to develop new hypotheses specifically about network theory".*
>
> *(Bender-deMoll and McFarland, 2006)*

We have seen in the previous chapter that producing layouts of single static networks is already challenging. Naturally, the task becomes even more difficult, if a coherent sequence of layouts for a dynamic network, i.e., an evolving sequence of graphs, has to be computed. This is because, in addition to the criterion of faithfully displaying structural properties, the sequence of layouts should convey the evolution of the network.

The goal is to enable the user to maintain an evolving cognitive model of the graph as it changes over time, which is referred to as the *mental map* (Eades, Lai, Misue, and Sugiyama, 1991). A major paradigm here is to largely retain those parts of a drawing in which, compared to the preceding one, little structural change occurred, so that the time a viewer spends on familiarizing with the preceding drawing is not wasted. The concept is easy to grasp intuitively, however, not trivial to formalize for algorithms. Additionally, trying to preserve the mental map is often in conflict with aesthetic criteria for the individual static representations.

The easiest (and, as will be demonstrated, effective) way to handle maintenance of the mental map is to force positions of vertices to be stable over time. Algorithmically, layout approaches for the static case have to be modified to strike the right balance between structural changes and positional stability. In our scenario of general, undirected graphs, several force-directed approaches have been proposed to address this trade-off, where stability is introduced by augmenting the method with additional forces. We will present variations of the stress minimization approach, where, in a similar way, the objective function is augmented with additional terms caring for stability.

Our variants are realizations of three general approaches to infuse explicit control for stability into general-purpose static layout algorithms. Maximum stability is achieved in *aggregation* approaches where fixed vertex positions are obtained from the layout of an aggregate of all graphs in the sequence. Alternatives are based on *anchoring* vertices to reference positions, or *linking* vertices to instances of themselves that are close in the sequence.

In the following, fundamental concepts with respect to dynamic graph drawing will be summarized, and related to previous work in the field. We then formulate generic realizations of the main approaches in the stress-minimization framework, and qualitatively illustrate their different behavior. Finally, an application to a larger real-world example will demonstrate practical effectiveness.

3.1. Fundamental concepts

Before we can formulate instantiations of stress minimization for addressing the dynamic graph drawing problem, some fundamental concepts related to this problem need to be clarified. The aim of the first part of this section is to give a conceptual overview. Some relations to previous work, together with more detail, are only presented later in Section 3.1.2 for sake of compactness.

The dynamic graph drawing problem Approaches for drawing dynamic graphs generally differ in three main aspects: the kind of representation they support (straight-line, layered, orthogonal, etc.), the class of graphs that can be drawn (trees, planar graphs, directed graphs, etc.), and the way in which stability is introduced. Regarding the first two aspects, we here are only interested in straight-line layouts of general graphs. In other words, we may position vertices freely, do not consider the routing of edges, and have no restrictions on the input graphs. More formally, the dynamic graph drawing problem addressed in this work can be formulated as:

Given: A sequence of T simple, undirected graphs

$$G^{(1)} = \left(V, E^{(1)}\right), \ldots, G^{(T)} = \left(V, E^{(T)}\right),$$

where the supergraph $\bar{G} = \left(V, \bigcup_{1 \leq t \leq T} E^{(t)}\right)$ is connected.

Find: A sequence of layouts $P^{(1)}, \ldots, P^{(T)}$ of two-dimensional positions $P^{(t)} = \left(p_1^{(t)}, \ldots, p_n^{(t)}\right)$, with $p_i^{(t)} \in \mathbb{R}^2, i \in V, 1 \leq t \leq T$, such that

- structural information of each individual graph $G^{(t)}$ is represented in a most effective way in $P^{(t)}$ (cf. Section 2.1),
- and at the same time, the understanding of the evolution of the network over time is facilitated.

Note, that we assume that the set of vertices remains the same for all graphs in the sequence. Additionally, we require that the supergraph $\bar{G}$, that is, the graph with the union of all individual edge sets as its edge set, is connected. This is to deal with the simplest problem instance, since if the supergraph is disconnected, the data can easily be split into the corresponding connected components, and layouts can be calculated independently for each of them.

Regarding terminology, there are different terms for the individual graphs of the sequence depending on the application context, like *time-slice*, *wave*, or *observation*. In the remainder, we will use *observation* or *individual graph*.

The first part of the problem statement corresponds exactly to the static layout problem for general graphs. Thus, it stands to reason that the use of the methods presented in the previous chapter, namely force-directed methods and, even more so, stress minimization, are appropriate for dynamic layout. But, how can we facilitate a user's understanding of the evolution of the network, and furthermore, integrate corresponding measures algorithmically?

The mental map When reading the drawing of a graph, a user will build an internal cognitive model of the data comprising positions, relations, orderings and groupings, both of individual vertices, and, on a higher level, of whole structures in a graph. This model is referred to as the user's *mental map* (Eades, Lai, Misue, and Sugiyama, 1991). Thus, the main goal to help the user in understanding of the evolution is to *maintain* the mental map and ease comparison between an observation and its predecessor in the sequence.

Figure 3.1 illustrates the case for a small dynamic network comprising 3 observations: The layouts in the left-hand column are obtained by applying stress minimization to each observation separately, without any measures to maintain the mental map. The right-hand column show layouts of the same network, obtained by an algorithm considering the mental map, which is presented later on. The stable arrangement of the right-hand drawings makes it easy to identify how and where structural changes occur in the network, while still being very readable. On the contrary, the left-hand side perfectly shows individual structure – especially in the sense of stress minimization – since shortest path distances are very well represented. However, without labeling (or in this case, color-marking), the interpretation of the evolution would be ambiguous, while with labeling, it surely is more demanding to trace structural changes exactly.

Indeed, enforcing *dynamic stability* on layouts, such that minor structural changes only result in minor changes in the graphical representation (Böhringer and Paulisch, 1990), is the core concept to address maintenance of the mental map. Technically, there are two ways to address dynamic stability in layout algorithms (Branke, 2001):

- Layout change is restricted to a subset of vertices, which are in the vicinity of a structural change;

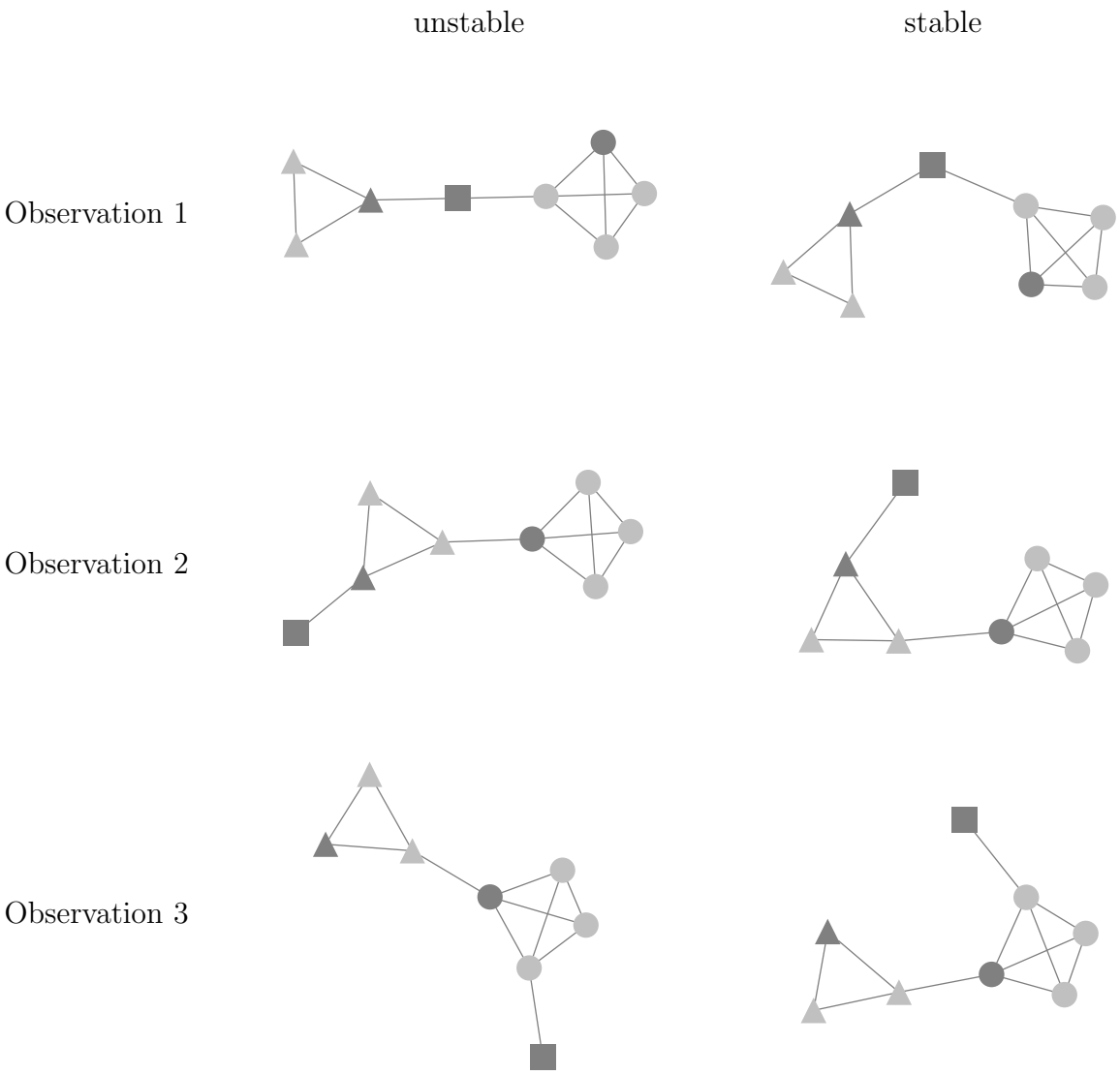


Figure 3.1.: Sample layouts of a dynamic network consisting of 3 observations. Not taking dynamic stability into account (left column) makes it harder to understand the evolution of the network.

- A difference metric that measures stability with respect to a reference layout (usually the previous in the sequence) is integrated in the algorithm to trade off individual layout quality with stability.

Branke (2001) surveys several difference metrics considered applicable for dynamic graph drawing in the literature: *absolute position*, *orthogonal ordering*, *proximity* and *topology*.[1] The idea behind each metric is to

- *Absolute position:* generally penalize absolute movement of vertices. This is measured by the sum of Euclidean distances between current and reference positions.
- *Orthogonal ordering:* preserve relative directions, i.e., if one vertex is left of another vertex in the reference, it should also be left of it in the current layout.
- *Proximity relations:* keep vertices close that are close to each other in the reference layout. One way to evaluate difference is by means of proximity graphs, that contain an edge between two vertices if those are deemed close by some proximity measure. Several instance of proximity graphs are plausible, e.g., the Gabriel graph, the relative neighborhood graph, or a Delaunay triangulation; see Jaromczyk and Toussaint (1992) for an overview. The layout preserves proximity if the proximity graph does not change.
- *Topology:* preserve the combinatorial information contained in the reference layout. Topology in this case refers to the circular arrangement of neighboring vertices and the regions (faces) incident to a vertex in the drawing. The measurement of change might be achieved by comparing the so called *dual graphs* (Di Battista, Eades, Tamassia, and Tollis, 1999) induced by each layout, which capture this combinatorial information.

Practically, algorithms for drawing general dynamic graphs, including our stress minimization variants presented later, are almost exclusively utilizing absolute position as a difference metric (cf. Section 3.1.2). This is justifiable, since, first of all, it is relatively straight-forward to implement by additional forces in force-directed methods, and as additional stress terms in a weakly constrained stress function. Second, absolute position can be regarded as a proxy for other metrics, because if vertex movement is constrained, changes in orthogonal ordering, proximity or topology are more unlikely.

Still, the question, whether the paradigm of restricting vertex movement really does help to better understand an evolving network, is a legitimate one. Qualitative illustrations, like Figure 3.1, where the right-hand representation follows this paradigm, and examples presented later on, speak favorably.

Several user studies have been conducted to evaluate the impact of preserving the mental map in this way. Although it has been shown in earlier studies (Saffrey and Purchase, 2008; Purchase and Samra, 2008) that representations incorporating means to maintain

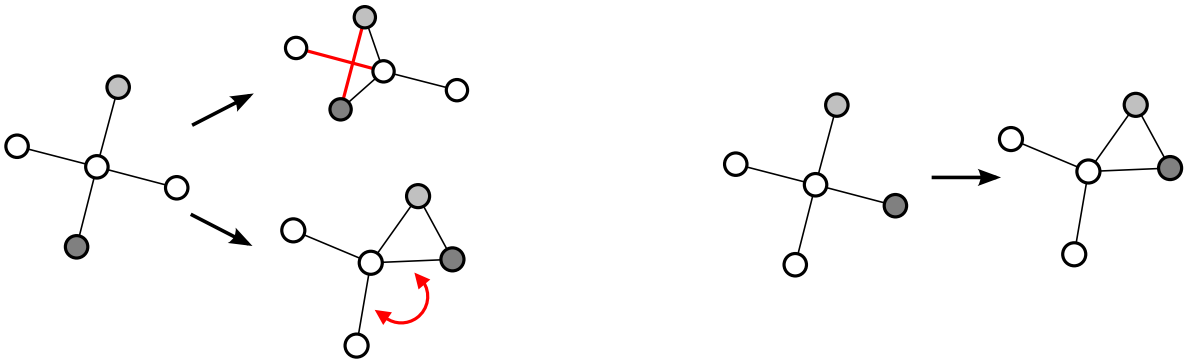
[1] Note that these and more have been studied for use as similarity measures of orthogonal drawings by Bridgeman and Tamassia (2000).

the mental map are generally preferred by users, they were not as promising: Counter-intuitively, there was no significant positive benefit of mental map preservation compared to a representation of independently calculated layouts of each observation. However, just recently, Archambault and Purchase (2013) found a significant positive effect of positional stability for tasks that require orientation in the evolving network. Specifically, users were to identify single vertices and paths in the last observation of a sequence, that were highlighted in the first observation only. The authors also address why previous studies might have been inconclusive. Those studies involved only a small number of independently moving vertices, and used preattentive highlighting throughout the sequence, both of which issues reduce the need for a consistent mental map. Furthermore, the tasks involved were more related to readability, like counting degrees, or determining lengths of shortest paths.

Logical and physical update Our problem formulation for dynamic graph drawing and the considerations regarding the mental map address one of two major tasks in visualizing a dynamic network: developing a so called *logical update* from one graph in the sequence to the next one, that is, determining geometric features, in our case, positions for vertices, for each observation in the sequence. The second task is developing a *physical update* of the results for the media on which the sequence is presented. In print media, the physical update is mostly restricted to a *small multiples* representation (Tufte, 1990), which depicts the sequence as a series of similar-sized graphics, often arranged in a grid-like fashion, as in Figure 3.1. With electronic media, however, it is also possible to use *animation* between observations.

Obviously, both logical and physical update need careful consideration to support maintenance of the user's mental map. However, we argue that, in the same way as meaningful geometry is paramount for a readable static graph visualization, even the most sophisticated physical update cannot remedy a poor logical update. The techniques presented in the remainder of this work exclusively deal with the logical update, yet, a few notes need to be taken about the physical update.

The discussion, whether a small multiples representation or an animation is more suited for displaying dynamic data, is very controversial. A general objection against animation is that it leads to larger cognitive overhead for the viewer. One of the harshest critics of animation are Tversky, Bauer Morrison, and Betrancourt (2002), who point out limitations of animation when used in the role of teaching complex systems. In the context of trend visualization, Robertson, Fernandez, Fisher, Lee, and Stasko (2008) study effectiveness of a small multiples presentation versus animation, and conclude that small multiples are more appropriate for analysis of the data, while animation is useful for presenting results. On the other hand, Griffin, MacEachren, Hardisty, Steiner, and Li (2006) find that animation is beneficial in identifying and tracking clusters in animated maps. The controversy persists for investigations specifically addressing visualization of dynamic networks. While the study of Farrugia and Quigley (2011) indicates that small multiples are generally more effective, Archambault, Purchase, and Pinaud (2011)



(a) unfortunate (b) fore-sighted

Figure 3.2.: Knowledge of future changes can inform the choice between otherwise equally good layouts.

conclude that small multiples are better suited for comprehension tasks, while animation is preferable for tracking sets of nodes and edges added to the graph. Furthermore, the recent study of Archambault and Purchase (2013) shows that animation is helpful for users to orient themselves in an evolving network.

If animation is used for the physical update, intermediate states between the two states of the logical update have to be determined. With layout algorithms that use iterative optimization, one could use the trajectories of vertices from the initial to the final position. In practice, however, linear or sinusoidal coordinate interpolation offer better control and are more commonly used to determine such trajectories. More sophisticated transition methods are described, for instance, in Friedrich and Eades (2002); Friedrich and Houle (2002); Nesbitt and Friedrich (2002).

Finally, developing the physical update for dynamic graphs is more than the mere choice between small multiples or animation. For example, representations may be enriched with highlighting or annotations, where, to our knowledge, no research exists specific to dynamic graphs beyond principles taken from general graphical design.

Online and offline scenarios Getting back to the logical update, two scenarios need to be distinguished, because they differ in the amount of information available to the algorithm when a layout is determined. Either, the entire sequence of graphs is known in advance, i.e., before any layout is required. This is generally referred to as the *offline scenario*, where all graphs are given to the algorithm together. Or, future graphs are unknown at the time that an intermediate state is to be laid out. This is referred to as the *online scenario*. Here, graphs are presented to the algorithm one at a time; however, the algorithm may have a memory with respect to earlier graphs and layouts of the sequence. In an online scenario, it is more difficult to maintain dynamic stability, because future changes are not known at the time that vertices have to be placed.

Let us illustrate this phenomenon by the small example in Figure 3.2 consisting of a 4-star in the first observation, and an additional edge between the two gray vertices in the second observation. In an online scenario, there is no preference for any of the two layouts of the first observation, whereas in an offline scenario we can actually choose the one that will lead to better quality and less movement.

In the remainder of this work, we will generally assume to be in an offline scenario, as is indicated by our problem definition. This is especially justifiable for networks from social science, as most often a sociological survey is completely executed over a period of time, and exploration and analysis are only performed afterwards.

3.1.1. Algorithmic integration of dynamic stability

Having set that positional stability is an adequate concept to care for coherence in layouts of dynamic networks, the question is how this can be integrated, algorithmically. Before we treat general approaches answering this question, we shortly review a useful tool to match two given layouts to maximize positional stability, without changing relative positioning in each layout.

Procrustes analysis When comparing two layouts, these may have large differences in position even if they are essentially the same, simply due to translation, reflection, rotation, or dilation. Naturally, this issue is not only related to dynamic graph drawing, but exists in many other domains, and the problem to remove these degrees of freedom is an important one to asses the goodness-of-fit between two configurations in MDS. The technique to match two configurations by translation, reflection, rotation and dilation such that the sum of squared positional differences is minimized, is known as *Procrustes analysis*. Although the technique was treated earlier, Sibson (1978) is the first to develop a consistent and unified algebra; a good overview is given by Cox and Cox (2001, Chapter 5).

In our context, we are only interested to find the best match under translation, reflection, and rotation, since dilation would interfere with the interpretation of unit edge lengths and shortest paths distances, respectively. Technically, a solution is obtained by performing two steps to match a two-dimensional layout $P \in \mathbb{R}^{2\times n}$ to another layout $Q \in \mathbb{R}^{2\times n}$, assuming all vertices appear in the same order in P and Q:

1. Standardize both P and Q to have their centroid at the origin, by subtracting mean vectors from each of the respective points.

2. Find the matrix $A = \left(P^T Q Q^T P\right)^{1/2} \left(Q^T P\right)^{-1}$ and rotate P to PA.

Note that Procrustes analysis can be implemented to run in linear time, since matrix operations from step 2 only involve 2×2-matrices in our case.

Naive approach The simplest (and most common) approaches to take stability into account algorithmically are based on variants of the spring embedders of Fruchterman and Reingold (1991) or Kamada and Kawai (1988), in which the iterative computation for each graph in the sequence is initialized with the preceding layout (Bender-deMoll and McFarland, 2006; Huang, Eades, and Wang, 1998; Geipel, 2007; Groh, Hanstein, and Wörndl, 2009).

An implicit assumption is that consecutive graphs are similar in general, so that the initial layout is not too far from a locally optimal one. The method is therefore easy to implement, more efficient than computing a layout from scratch, and applicable in both on- and offline scenarios.

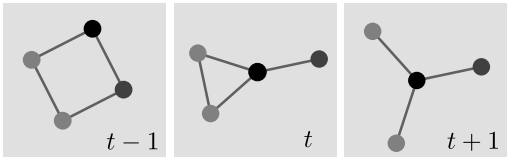
In spite of the convenient properties enumerated so far, the approach is rather problematic, because it does not address stability in a controlled way and may hence result in excessive and unnecessary movement of vertices. This can be seen from the left-hand side of Figure 3.1, where layout computation has only been initialized by the previous layout. Moreover, by failing to make use of the existing knowledge about the future in offline scenarios, the approach is biased towards earlier configurations, and therefore prone to suffer from poor local minima precisely as already illustrated in Figure 3.2. Hence, layout quality tends to degrade over the course of the sequence.

Explicit control for stability As mere initialization is not enough to control stability in layout algorithms for the static case, they have to be modified to strike the right balance between structural changes and positional stability. The trade-off between readability and stability is formalized in Brandes and Wagner (1997a) and a similar principle for offline scenarios is proposed in Diehl and Görg (2002). In both, the idea is to augment the static algorithm with stability criteria, and jointly optimize both quality and stability.

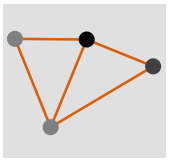
Current proposals generally take one of the following three approaches, which are illustrated in Figure 3.3.

Aggregation. All graphs in the sequence are aggregated into a single graph that has one vertex representing its instances in all observations. In its simplest form, this is the supergraph having the union of the observation's edge sets as its edge set. For the layout of each observation, vertices inherit the position of their representative from a (static) layout of the aggregated graph. Thus, a vertex' position remains completely fixed throughout. This approach is used, e.g., in Brandes and Corman (2003); Dwyer and Gallagher (2004); Moody, McFarland, and Bender-deMoll (2005).

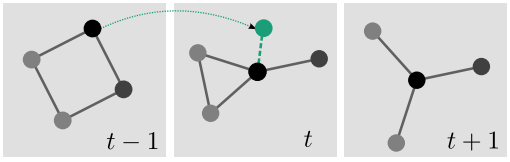
Anchoring. Vertices are connected to immobile copies fixed to a desired location which may be, for instance, the previous position in an online scenario, or a reference position in an offline scenario. The layout algorithm strives to keep this connection short. Layouts of observations are calculated one by one. This approach is



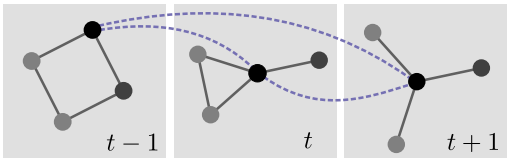
(a) Sequence



(b) Aggregation



(c) Anchoring



(d) Linking

Figure 3.3.: Different conceptual approaches to dynamic graph drawing.

used, e.g., in Lyons, Meijer, and Rappaport (1998); Brandes and Wagner (1997a); Frishman and Tal (2008); Boitmanis, Brandes, and Pich (2008).

Linking. All graphs in the sequence are combined into a single graph keeping each observation and all vertex instances of the same vertex intact. A connection is created between vertex instances representing the same vertex in other observations. Again, the layout algorithm strives to keep these connections short. A layout of this graph directly yields positions for all vertex instances in the sequence. This approach is used, e.g., in Dwyer and Gallagher (2004); Erten, Kobourov, Le, and Navabi (2004c); Dwyer, Hong, Koschützki, Schreiber, and Xu (2006a); Leydesdorff and Schank (2008).

3.1.2. Related work

In the following we will give a short history of work relevant for dynamic graph drawing, detail specific individual contributions and report earlier approaches that lead to the categorization of techniques just stated.

Layout adjustment and incremental layout Some of the earliest work in the context of dynamic graph drawing is not directly addressing the problem of finding suitable layouts for given sequences of graphs, but two related problems. The first one is *layout adjustment*, where the goal is to improve readability of a layout while maintaining the mental map, for example, to improve node distribution or removing node overlaps. The second is *incremental layout*, where a layout has to be updated in a stable way after a change has occurred in the graph, for example, after a manual change of a user in a graph drawing tool.

The single most recurring work related to dynamic graph drawing is the article of Misue, Eades, Lai, and Sugiyama (1995). Two algorithms are proposed for a layout adjustment scenario: one is to update a layout to avoid node overlaps, the second is about updating a layout in a focus and context view after the user has changed his focal point. Both techniques aim to preserve what the authors established to be named the user's *mental map*. The article probably is most famous for the attempt to conceptualize the intuitive phenomenon of change in a layout more precisely. The three mathematical models presented are *orthogonal ordering*, *proximity relations* and *topology*, which measure the extent of change. Note, however, that both the term "mental map" and similar models conceptualizing it are already presented by the same authors some years earlier in Eades, Lai, Misue, and Sugiyama (1991).

Böhringer and Paulisch (1990) are among the first to actively deal with maintaining a users mental map in an incremental layout scenario. They augment a well-known hierarchical layout algorithm for directed acyclic graphs (Sugiyama, Tagawa, and Toda, 1981) to incorporate *structural* and *dynamic* stability constraints. Here, structural constraints refer to user-specified constraints regarding absolute or relative position of vertices or clusters. The more important part with respect to our work is *dynamic stability*, which "is concerned with minimizing the difference between successive layouts of the graph", e.g., after the graph has changed and a new layout has to be calculated. They state that "ideally, making a minor change in the graph's structure should cause only a minor change in the layout". To achieve this, vertices are only allowed to change their level in the hierarchical layout, if they are in the *vicinity* of a change; a paradigm recurring in the literature.

The term *anchored* graph drawing was coined by Lyons (1992). In a layout adjustment scenario, her goal is to improve readability by distributing vertex positions in dense regions of a graph, while keeping vertices close to their initial positions. Four different approaches are presented. The first one, *geographic force*, is directly applicable to the more general dynamic graph drawing problem, and now established as a standard approach. In a spring embedder model, attractive forces are added between vertice's current locations and their initial position, i.e., anchored to those positions. The other three approaches are specific to layout adjustment: *Voronoi cluster busting* iteratively moves vertices into the center of Voronoi regions, *X-Y-ordering* moves vertices into the center of grid cells determined by the initial orthogonal ordering, and the *Fish-eye* approach applies a fish-eye function to improve distribution in dense regions. A later follow

up article of Lyons, Meijer, and Rappaport (1998) details the algorithmic concepts of the geographic force and Voronoi-based approaches and, moreover, provides a comparative evaluation using dedicated measures for distribution and similarity. More details will be given in Section 4.3.2.

North (1996) builds on the work of Böhringer and Paulisch (1990) and addresses incremental hierarchical layout, but more importantly, provides a general problem statement for dynamic graph drawing. Given a sequence of graphs, the goal is to find a "good" sequence of layouts, in the sense that users should be able to retain a persistent mental map, and graphical updates should reflect actual changes in the data. Moreover, North proposes a natural and generally acknowledged order of importance for desirable characteristics:

1. Consistency, or adherence to layout style rules – a layout algorithm should reveal given properties of interest. If, for example, the rule is to obtain a planar drawing for a planar graph, then an algorithm that produces a non-planar drawing of a planar input graph due to adherence to other properties is inane.

2. Stability – the algorithm should follow the principle of least change between successive layouts, however, subject to consistency. Interestingly, this implies that fundamental changes in the underlying graph structure actually should cause fundamental changes in a layout.

3. Readability – the layout produced by the algorithms should be easy to read, usually achieved by optimizing aesthetic criteria, see Section 2.1.

Regarding stability, North assumes that key characteristics are position and order relations in successive layouts. Moreover, he suggests that vertex stability is more important than edge stability, since the vertices are learned sites, and edges are traced on the fly. Another interesting concept that is mentioned is that of "age" or "memory": If a vertex has been updated recently or is in the vicinity of recently updated vertices, it may be a better candidate for a current update. However, this concept is mostly addressed implicitly in later work, if at all.

Generalized frameworks After the general problem formulation of dynamic graph drawing and terms related to it have been set, two important articles focus on generalizing concepts and formalizing frameworks to address the trade-off between individual layout quality and dynamic stability.

Brandes and Wagner (1997a) build upon a formalization of graph layout via random field models (Brandes and Wagner, 1997b). Essentially, a static layout model is described by a probability function $P(X = x)$, which expresses conformance of a given layout[2] x to given layout goals by a single random variable X. The higher the probability, the "better" a layout is. Given a sequence of graphs $G_1, \ldots, G_t$ and corresponding layout

[2]We use the original notation here to avoid confusion with the notation of a probability function.

models in a dynamic setting, one is interested in maximizing the joint probability $P(X_1 = x_1, \ldots, X_t = x_t)$. A natural way to incorporate dependencies between layouts into the joint probability is to formulate it by means of conditional probabilities, i.e.,

$$P(X_1 = x_1, \ldots, X_t = x_t) = \prod_i P(X_i = x_i | X_{<i} = x_{<i}),$$

where $X_{<i} = x_{<i}$ is shorthand for $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$. The authors call $P(X_i = x_i | X_{<i} = x_{<i})$ the *dynamic layout model* of G_i, and show that, assuming an online scenario where all previous layouts $x_{<i}$ are given, the goal is to maximize this conditional probability. The key insight now is that optimization of this dynamic model can be broken down into joint maximization of the static model and a *stability model*, since, by Bayes' rule, $\max P(X_i = x_i | X_{<i} = x_{<i})$ is proportional to

$$\max \underbrace{P(X_{<i} = x_{<i} | X_i = x_i)}_{\text{stability model}} \cdot \underbrace{P(X_i = x_i)}_{\text{static model}}.$$

This formalization reinforces the common sense approach to develop a dynamic layout algorithm by integrating stability criteria to a static layout design.

Furthermore, the authors demonstrate how to apply their framework to dynamic layout with spring embedders and with an algorithm for orthogonal layout. In both cases, proposed stability models directly transfer into each respective algorithmic concepts. For spring embedders, one stability model is *anchoring* which is based on a normal distribution around the previous vertex position. This model translates to an additional anchoring force similar to the ones in Lyons (1992). A second model proposed is *stiffening* which reinforces relative positions between vertices that remain connected.

Conceptually similar, Diehl and Görg (2002) formulate a framework for offline dynamic graph drawing via joint optimization of quality and stability. The problem formulation builds on metrics measuring quality and instability (the latter are referred to as *mental distances*): Given a sequence of graphs, the goal is to compute layouts such that the sum over mental distances between successive layouts is minimal, and the sum of quality measures over layouts is maximal. To achieve algorithmic feasibility, the authors relax the problem formulation; instead of requiring minimal mental distance, they only require that the distance metric between successive layouts must be smaller than a *tolerance value* δ.

The authors provide a generic algorithm for optimizing this relaxed problem. Each layout is initialized with a global layout; this might be a layout of just the *supergraph*, i.e., the graph having the union of individual vertex and edges as its respective sets, or of a more complex form of aggregation (Diehl, Görg, and Kerren, 2001). Hence, mental distance between successive layouts initially is optimal. Subsequently, an adjustment strategy is applied that trades off stability for quality, but is restricted to conform to the tolerance threshold. Four adjustment strategies are presented that differ in the kind of graphs to which individual graphs are compared with respect to mental distance:

- *Independent:* Only mental distance between current layout and global layout is taken into account. Yields the global layout for tolerance value $\delta = 0$.
- *Predecessor dependent:* Only mental distance between preceding and current layout is considered. This corresponds to anchoring when mental distance is Euclidean distance to the preceding position.
- *Context dependent:* Additionally to the preceding layout, mental distance to the layout of the subsequent graph induced by the global layout is taken into account.
- *Simultaneous:* The adjustment step considers all predecessor dependencies simultaneously. With Euclidean distances as stability metric, this yields a linking approach.

Technically, the authors realize their framework with a force-directed approach using simulated annealing; however, only little detail is given.

Further work In the following, several more recent variations of force-directed methods for dynamic graph drawing are presented. These corroborate the earlier categorization of approaches with explicit control for stability into aggregation, anchoring, and linking.

In the context of dynamic discourse networks, Brandes and Corman (2003) present an aggregated layout approach. Using the Kamada-Kawai algorithm, the aggregate layout is built by starting computation with a prefix of the sequence, and successively adding later observations one by one and executing a few iterations of the algorithm. The resulting complete aggregated layout is then depicted in a 2.5D visualization, where time is represented in the third dimension, by stacking images of individual observations on top of each other.

Dwyer and Gallagher (2004) present a similar 2.5D approach for visualizing changes in fund manager holdings, which they call *stratified* or *columns* layout since every vertex has exactly the same position in each layer. The approach builds upon an earlier work of Dwyer and Eades (2002), where also a variation of the columns layout is suggested. Here, the columns are allowed to bend to show temporary closeness of vertices. Practically this is realized by introducing stiff springs between instances of same actors in consecutive observations, i.e., linking them. Unfortunately, little algorithmic detail is given.

Erten, Kobourov, Le, and Navabi (2004c) provide two more instantiations of an aggregation and a linking approach for force-directed methods. The aggregate network is composed as the union of vertices and edges, however, both receive weights corresponding to their respective lifetimes. Attractive forces are reformulated to respect the edge weights, such that vertices adjacent in many observations are placed more closely to each other. Additionally, a new force connecting each vertex to the center of the drawing depending on its vertex weight is employed, such that persistent vertices should remain in the center of the drawing, while more volatile ones appear in the periphery. In the linking approach (which the authors call *merged* layout) either consecutive or all

instances of a vertex are connected by virtual edges, and given a global weight determining stability. Then the whole system is solved simultaneously. More implementation details, especially with respect to formulation of forces and realization as a multi-level algorithm, are given in Erten, Harding, Kobourov, Wampler, and Yee (2004b). An application of the linking approach to bibliographic networks is presented in Erten, Harding, Kobourov, Wampler, and Yee (2004a).

Note that, in the same work, Erten, Kobourov, Le, and Navabi (2004c) propose an approach relying on initialization only, called *independent iterations*, for a dynamic scenario comprising only two graphs. Employing multiple runs of a standard force-directed minimization, they suggest to initialize layout computation of each graph with the last intermediary result of the other graph, until either the sum of pairwise Euclidean distances is below a threshold or a maximum number of iterations is performed. However, similarly to the naive approach of just initializing with the preceding layout, this approach offers no explicit control for stability.

Another interesting application of a linking approach is given by Dwyer, Hong, Koschützki, Schreiber, and Xu (2006a), where the goal is to visualize several centrality measures of the same static network. Although not dealing with dynamic data, the problem is related: In a 2.5D visualization of the network, where each centrality is shown in a separate layer containing a centrality drawing similar to the one shown in Figure 2.7b, vertex instances of the same vertex should be as close as possible. Employing a Fruchterman-Reingold algorithm, the suggested solution is to augment the series with additional forces linking vertex instances in consecutive layers and solve the whole system simultaneously.

Compared to the basic anchoring approaches from Lyons (1992) and Brandes and Wagner (1997a) that introduce an explicit anchor force, a more involved approach is presented by Frishman and Tal (2008). Besides employing a multi-level force-directed layout algorithm and utilizing the GPU to address efficiency, the key element in this work is to control stability by allowing movement of vertices only with respect to actual structural change between successive observations. The degree of flexibility is determined by *pinning weights*: These are high if the neighborhood of a vertex has not changed compared to the last observation, and are lowest for vertices that are directly incident to a change. During execution of the force-directed algorithm, only vertices with a pinning weight lower than a certain threshold are allowed to move, while this threshold increases with the number of iterations. In this sense, this approach could be classified as binary anchoring, since only low-weight vertices are free to move, while high-weight ones are completely anchored.

Lastly, there are two previous articles, where stress minimization has been adapted to obtain drawings of dynamic graphs. Boitmanis, Brandes, and Pich (2008) visualize the evolution of the physical structure of the Internet. Besides proposing several techniques to handle the visual complexity of this large network, they present a stress function augmented by a stability term that anchors each vertex to its position in a previous drawing. Leydesdorff and Schank (2008) employ a stress-based linking approach to

asses interdisciplinarity among several journals by visual analysis. Although similar, our instantiations of stress minimization, that are presented next, are more general than these two proposals.

3.2. Dynamic variants of stress minimization

In Chapter 2, we have argued that stress minimization is a very robust and effective tool to obtain static layouts for general graphs. It thus stands to reason that we will benefit from its properties in the dynamic scenario, too. In the following, we will present variants of stress minimization realizing the general aggregation, anchoring and linking approaches, discuss differences between those qualitatively, and demonstrate practical effectiveness.

3.2.1. Aggregation

Maximum stability is obtained when a vertex maintains its position throughout the entire sequence of diagrams. This is called the *flip book* approach in Moody, McFarland, and Bender-deMoll (2005). Given a sequence $G^{(1)} = (V, E^{(1)}), \ldots, G^{(T)} = (V, E^{(T)})$ of T graphs, $1 \leq t \leq T$, we are to determine layouts $P^{(1)}, \ldots, P^{(T)}$ such that a vertex is placed at the same position throughout. In other words, we are looking for one layout P for the vertices in V and let $P^{(t)} = P$ at all times $t = 1, \ldots, T$.

Naive approach The simplest way to obtain fixed layout positions $\bar{P}$ is to compute a layout for the supergraph $\bar{G}$,

$$\bar{G} = \left(V, \bigcup_{1 \leq t \leq T} E^{(t)} \right),$$

by applying static stress, i.e., computing shortest-path distances in $\bar{G}$, and performing stress minimization after initializing by classical scaling. We will denote this approach with $\mathsf{stress}^{\bar{G}}$.

A problem with this simple approach is that it does not take into account varying lifetimes of connections. Unless the changes in the sequence are minimal, the supergraph generally is much more dense than individual observations, thus the static layout of the supergraph will likely be cluttered, and otherwise distinguishable groups likely be diffused by ephemeral connections. Previous work attempts to remedy this by either partitioning the supergraph into smaller graphs where entities have the same lifetimes (Diehl, Görg, and Kerren, 2001), or weight its edges proportional to their lifetimes (Erten, Kobourov, Le, and Navabi, 2004c).

Aggregating distances We argue that, with stress minimization, suitable positions can be obtained by applying stress minimization to an aggregation of distances in the input sequence. Let

$$D^{(t)} = \left(\delta_{ij}^{(t)}\right)_{i,j\in V}$$

denote the matrix of shortest path distances of observation t. We combine all shortest path information into a single stress function using the mean shortest path distance per dyad, and use the same position variables p_i for each instance $i^{(t)}$ of the same vertex i,

$$\mathsf{stress}^{AGG}(P) = \sum_{i<j} \bar{\omega}_{ij} \left(\bar{\delta}_{ij} - \|p_i - p_j\|\right)^2 , \tag{3.1}$$

where $\bar{D} = \left(\bar{\delta}_{ij}\right)_{i,j\in V}, \bar{\delta}_{ij} := \frac{1}{T}\sum_{t=1}^{T} \delta_{ij}^{(t)}$, contains the mean shortest-path distances.

We define weights $\bar{W} = (\bar{\omega}_{ij})_{i,j\in V}$ as

$$\bar{\omega}_{ij} = \frac{1}{\bar{\delta}_{ij}^2} \cdot \frac{1}{1 + \textsc{var}(\delta_{ij})} ,$$

where $\textsc{var}(\delta_{ij}) := \frac{1}{T}\sum_{t=1}^{T} \left(\delta_{ij}^{(t)} - \bar{\delta}_{ij}\right)^2$ is the variance of distances within a dyad across all observations. Thus, representation accuracy of dyads that are connected via short paths most of the time is emphasized. By additionally scaling with the variance, priority is given to structures that are relatively stable throughout the sequence.

Computation Since, technically, the aggregate stress function consists of the same type of terms as before, we can also use the same algorithms for its minimization: Layout computation is initialized by classical scaling. Subsequently, $\mathsf{stress}^{AGG}(P)$ is reduced via majorization.

As with static stress, computational complexity is asymptotically dominated by calculation of shortest paths, taking $\mathcal{O}(Tn\bar{m})$ time, where $\bar{m} = \left|\bigcup_{1\le t\le T} E^{(t)}\right|$. Calculation of distance means and variances takes $\mathcal{O}(Tn^2)$ time, while classical scaling and stress majorization run in $\mathcal{O}(n^2)$, assuming the involved number of iterations until convergence can be regarded as constants.

Infinite distances Even though we require the supergraph to be connected it may still be the case that some individual networks of a sequence contain multiple disconnected components, resulting in infinite distances and undefined weights. Being in an offline scenario, other observations can be used to fill in gaps as follows. An infinite distance in a dyad is replaced by interpolating between the two finite distances observed previously and next for this dyad, and by adding a small constant, say 1, to emphasize temporary disconnectedness. Let t_{ij}^{-} be the most recent and t_{ij}^{+} be the next observation in which

actors i and j have finite distance $\delta_{ij}^{(t_{ij}^-)}$ and $\delta_{ij}^{(t_{ij}^+)}$. Then the interpolated distance for dyad $\{i, j\}$ is

$$\delta_{ij}^{(t)} = \left(1 - \beta_{ij}^{(t)}\right) \delta_{ij}^{(t_{ij}^-)} + \beta_{ij}^{(t)} \delta_{ij}^{(t_{ij}^+)} + 1 \ ,$$

where $\beta_{ij}^{(t)} = (t - t_{ij}^-)/(t_{ij}^+ - t_{ij}^-)$. In the special cases that there is no last or next finite distance, we do not interpolate, but use the one existing finite distances distance plus the same constant. Although unlikely, it may happen that a dyad has infinite distances at all times, even if the aggregated graph is connected. In this case, a sufficiently large distance Δ with a small weight Ω is used. We suggest $\Delta := \sqrt{n}$ and $\Omega := 1/n$. This is in analogy to the height and width of an equally spaced grid with n points.

The complexity of calculating replacements of all possible infinite distances in this way is in $\mathcal{O}(Tn^2)$ and, thus, does not affect the overall complexity.

Discussion We will next discuss qualitative differences of variant approaches using a small example sequence comprising 4 observations, shown in Figure 3.4. This example will also be used for later discussions. The network consists of 10 vertices, with a group of 4 circular, a group of 3 triangular and a group of 3 square vertices. While edges within the circular and the triangular groups remain relatively stable (black lines indicate edges remaining throughout all observations), the square group is only completely connected in the first observation, whereupon the dark square vertex disconnects from this group and remains distant to the other square vertices. Note also, that the triangular and circular group are connected through varying dyads throughout the sequence.

The first two columns show layouts without explicit control for stability. For both columns, subsequent layouts are aligned by Procrustes analysis.

The first column shows static stress layouts of each individual observation, with initialization by CMDS. We observe that each individual graph has a very clear structure, and that stress minimization can represent graph distances very well. However, there is also large movement of vertices in between observations. For example, the dark gray vertices must move from the center in Observation 1 to the periphery in Observation 2. Conversely, the white circular and triangular vertices start in the periphery, and are directly connected in Observation 2 and Observation 4. This volatile connection also creates movement, and more severely, results in flips in the combinatorial embedding[3] of the triangular and circular groups. The triangular group flips in Observations 3 and 4, while the circular group flips in Observations 2 and 3.

In the second column, layout computation is initialized with the layout of the previous observation. The naive approach in this case does not reduce movement at all, but only avoids the flips of the triangular group.

[3]The combinatorial embedding corresponds to the cyclic order of edges incident to the same vertex.

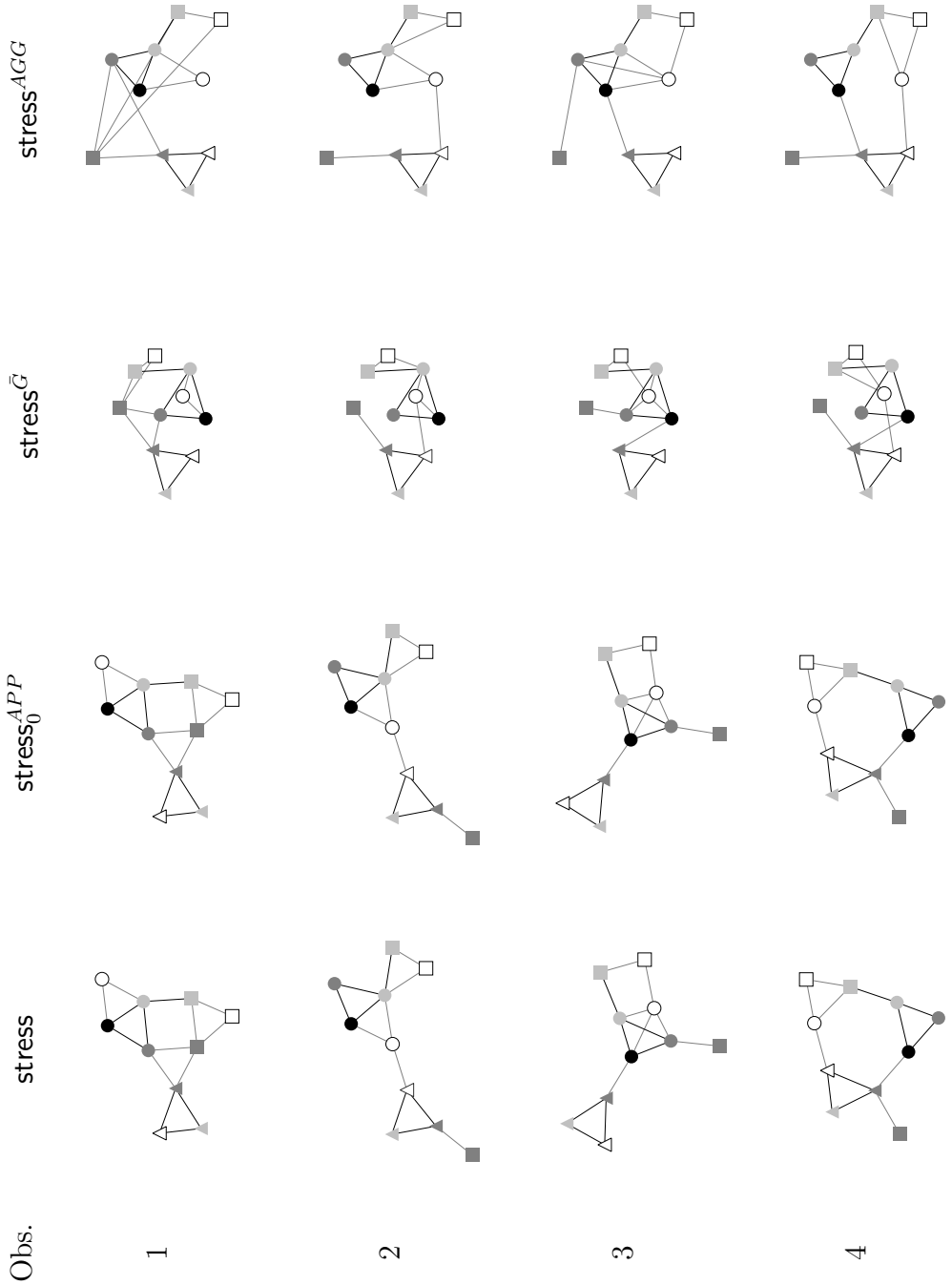


Figure 3.4.: Qualitative Example. Layout by static stress produces very readable layouts, but also involves large movement. Initializing by the previous layout (stress_0^{APP}) essentially yields the same layout. A static stress layout of the supergraph $\bar{G}$ eliminates movement, but does not reveal distinct structure, and looks cluttered. Aggregate stress remedies those problems.

The last two columns show layouts obtained by aggregation, where the third column shows the layout obtained by static stress applied to the supergraph (the naive approach), while the fourth column shows the layout obtained by applying stress^{AGG} as defined in Equation 3.1.

Both naturally exhibit perfect stability, that is, there is no movement of vertices at all. However, the naive approach does not reveal the distinct structure in each observation well. The main culprit here is the dark square vertex, which is only connected to the other square vertices in the first observation, but distant to them in the later ones. Because the naive approach does not treat such volatile connections, we generally get a cluttered impression, which would remain even if layouts would be scaled to match the size of layouts shown in the first two columns.

In contrast, stress^{AGG} clearly separates stable subgroups, and thus is able to faithfully represent global structure, due to consideration of lifetimes and variance in distances. While the layout of the first observation does not match the natural arrangement, the long edges actually reveal that the connections of the dark square vertex are outliers as seen from a global perspective. Furthermore, due to the focus on representation accuracy on stable distances, the individual structure in Observations 2 to 4 is still visible, albeit arranged slightly unnatural; see, for example, the dark gray vertex in Observation 3.

3.2.2. Anchoring

The main idea of the Bayesian approach to online dynamic graph drawing (Brandes and Wagner, 1997a) is an explicit modeling of the trade-off between layout quality as measured by an objective function, and layout stability with respect to the previous drawing as measured by a difference metric (Bridgeman and Tamassia, 2000). Stress minimization can be naturally adapted to this approach, where the static model's objective function is stress, and the stability model is evaluated by positional deviation to reference positions.

Anchored stress Following the general approach for weakly constrained stress minimization (cf. Equation 2.13), we formulate anchored stress as

$$\mathsf{stress}^{A}_{\alpha}\left(P^{(t)}\right) = (1-\alpha)\cdot \underbrace{\mathsf{stress}\left(P^{(t)}\right)}_{\text{individual layout quality}} + \alpha \cdot \underbrace{\sum_{i\in V} \phi_i^{(t)} \left\| p_i^{(t)} - p_i \right\|^2}_{\text{stability}}, \tag{3.2}$$

where $P = (p_i)_{i\in V}$ denotes the reference layout, and weights $\phi_i^{(t)}$ allow for inter-vertex variation in deviation tolerance.

The stability term thus corresponds to a point-wise penalty for deviations from the reference layout, and the parameter $0 \leq \alpha \leq 1$ provides explicit control of the trade-off

between quality (original stress) and stability. Note that minimizing stress^A_α for $\alpha = 0$ corresponds to regular stress minimization without control for stability, and $\alpha = 1$ yields the reference layout, since no deviation is tolerated.

For now, we use constant stability weights $\phi_i^{(t)} := 1$ for all i and t. More sophisticated choices, however, may be useful to compensate for cases with highly varying degrees or localized structural change. Another potential use of stability weights is as normalizing factors, such that the quality and stability part in stress^A_α are on equal scales, and thus, parameter α can be interpreted more easily in between its extreme values. More technical and user-oriented experimentation is needed, though, to quantify dependencies on these parameters.

Reference layout and initialization In an online scenario, the layout of the previous graph in the sequence is used as reference, since – due to lack of knowledge of future observations – this is the best choice to accommodate coherence of consecutive layouts. When the previous layout is used as reference, stress^A_α is a direct translation of the online Bayesian and associated force-directed anchoring approaches to stress minimization.

However, as we assume an offline scenario, knowledge of the whole sequence can be introduced to anchoring by using the aggregate layout just described as reference. The latter serves as a baseline for representing stable overall structures, and thereby facilitates the formation of a persistent mental map. In this case, stress^A_α can be seen as a weakly constrained version of the independent variant of the framework of Diehl and Görg (2002), allowing deviations from aggregate positions if they lead to improved representation of momentary structures for each individual graph in the sequence.

Regarding initialization of individual layout computation, a natural consideration is to use the layout of the preceding observation, as an additional measure to improve stability besides modeling it explicitly as a layout objective. Assuming that consecutive observations are structurally similar, stress minimization is expected to reach a similar local minimum as for the preceding observation. Hence, ambiguities will be resolved in favor of the preceding layout if deviations from the reference layout must occur due to structural changes. For the same reason we may also expect that only few iterations are needed to compute a layout. On the other hand, one can argue to use the classical scaling solution for initialization, to emphasize the aspect of individual quality by guiding optimization towards a local minimum similar to the one obtained by static stress.

Depending on initialization and the type of reference, we thus obtain four anchoring methods. The first two are purely online, whereas the second two incorporate offline information by means of using the aggregate layout as reference:

- **APP** initialize with previous layout (classical scaling for the first network), and also anchor to previous layout (no anchoring for the first network).
- **ACP** initialize with classical scaling, anchor to previous layout (no anchoring for the first network).

- **APA** initialize with previous layout (aggregate layout for the first network), anchor to aggregate layout stress^{AGG}.
- **ACA** initialize with classical scaling, anchor to aggregate layout.

Computation Since the stability term corresponds to a regular stress term measuring squared deviation of Euclidean distance from the desired zero distance, algorithms for minimizing stress can be used without alteration, and convergence properties hold. To reduce artifacts originating from translations and rotations of the reference with respect to the initial layout, these are aligned by Procrustes analysis before minimizing stress. After a layout has been computed for each observation, we post-process consecutive layouts by Procrustes analysis again, to ensure maximal positional stability without change of relative distances in each individual observation.

With localized stress minimization, the sum of votes for each vertex i receives one additional term in the inner loop of Algorithm 1 (see Section 2.3). Thus, complexity of this part is the same as for static stress, and sequential minimization of all observations takes $\mathcal{O}(Tn^2)$ time. The costs for computing the various initial and reference layouts, copying initial positions, and performing Procrustes analysis clearly do not increase the complexity.

Discussion We will first discuss anchoring variants using moderate anchoring. Figure 3.5 shows layouts of the small example used earlier, obtained by variant anchoring approaches for a relatively low trade-off parameter $\alpha = 0.2$. In the first two columns, the reference layout is the previous one, while in the last two columns the reference layout is the aggregate layout.

Regarding anchoring to the previous, both variants naturally start with the exact same layout for Observation 1 as computed by static stress as shown in Figure 3.4, since both are initialized by CMDS, and since there are no stability terms for each first observation. Due to the stability term contributing from Observation 2, the unfolding of, especially, the square vertices is slowed, which is most clear in Observation 3, thus reducing movement. Compared to the static stress layouts, the anchors to the previous positions avoid the large vertical movement of the whole structure in Observation 4, but this comes at the cost of unnatural arrangement of the dark square vertex, due to its bias toward its previous placement.

Similar to static stress and static stress with initialization by previous, initialization does not have a major influence on these layouts for this small example. Like before, initialization by previous avoids the flip of the triangular group in Observation 3, which however now results in a flip from Observation 3 to Observation 4. Although moderate anchoring to previous positions reduces movement, we still observe the exact same flips in the embedding of the circular group as with static stress.

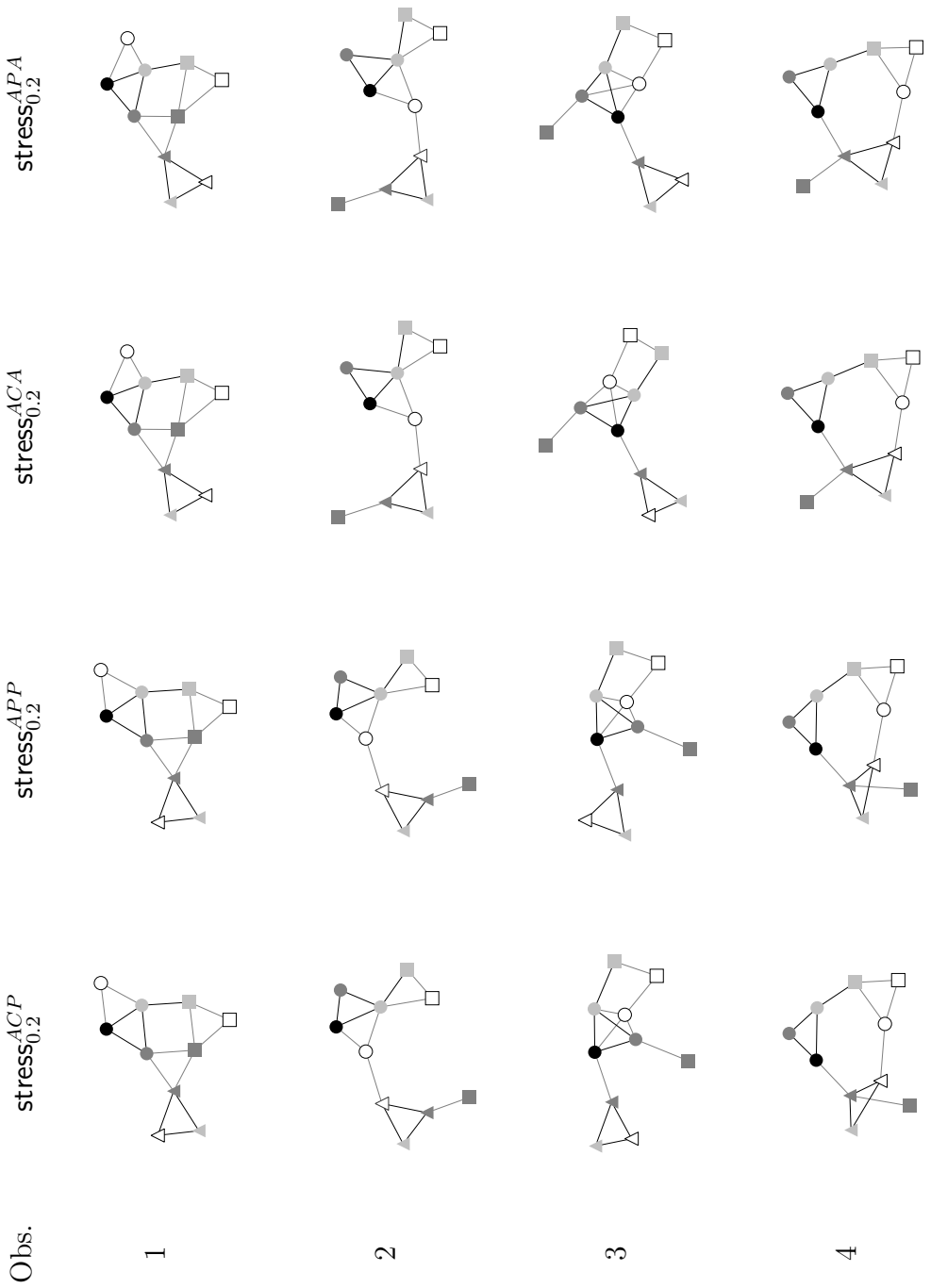


Figure 3.5.: Qualitative Example. Layout by moderate anchoring. Anchoring to the aggregate layout is beneficial for later observations, but is more severely affected by initialization. Anchoring to aggregate with initialzation by previous performs best.

Let us now discuss anchoring to aggregate positions. Due to only moderate anchoring, the qualitative part of the stress function still has a large influence on the result, which can be seen in Observation 1, where the layout is similar to the static one, and differs largely from the reference (cf. Figure 3.4). Still, the effect of the stability term is high enough to place the dark square vertex differently in later observations corresponding to its position in the aggregate layout, and regardless of initialization. This does not increase movement, but has a beneficial effect for quality in later observations, especially in Observation 4, which yields a more faithful representation of the structure.

In contrast to anchoring to the previous, anchoring to aggregate seems more severely affected by initialization. While initialization by previous removes all flips in the embedding but the flip in the circular group from Observation 1 to Observation 2, initialization by CMDS still produces the flips in the triangular group. Moreover, the latter introduces an unnatural flip of the light-gray and white circular and square vertices in Observation 3 regardless of their position in the reference layout. This can only be explained by the fact that each two of those having the same shape have exactly the same position in a CMDS layout due to structural equivalence, and hence, an unfavorable order of processing in the local update of stress minimization could result in such artifacts. Note, that this phenomenon disappears with stronger trade-off parameter (for this example with $\alpha \geq 0.23$).

Overall, among anchoring approaches with moderate anchoring, stress^{APA} produces the most favorable result.

Next, Figure 3.6 shows the results of the four anchoring variants with strong anchoring ($\alpha = 0.6$). We first observe that the choice of initialization now does not visibly affect the result. For both choices of reference, movement is largely reduced. Mainly, the vertices with volatile connections, that is, the dark square and white circle, exhibit evident movement. However, the bias towards the reference layout is now very visible, as expected. In our example, where the first observation is actually an outlier with respect to global structure, the online approaches are severely affected, such that the clear individual structure of Observation 4 can hardly be recognized anymore. By taking into account offline information, quality of layouts by anchoring to the aggregate is only affected in this outlier case, while still producing readable layouts for the other observations.

3.2.3. Linking

The main idea of the linking approach is to implicitly make use of all information about the networks of a sequence in an offline scenario. All instances of the same vertex are *linked* with each other, so as to stabilize their positions throughout the sequence. In contrast to the anchoring approach, layout calculation is not performed one after each other, but the whole system is computed simultaneously.

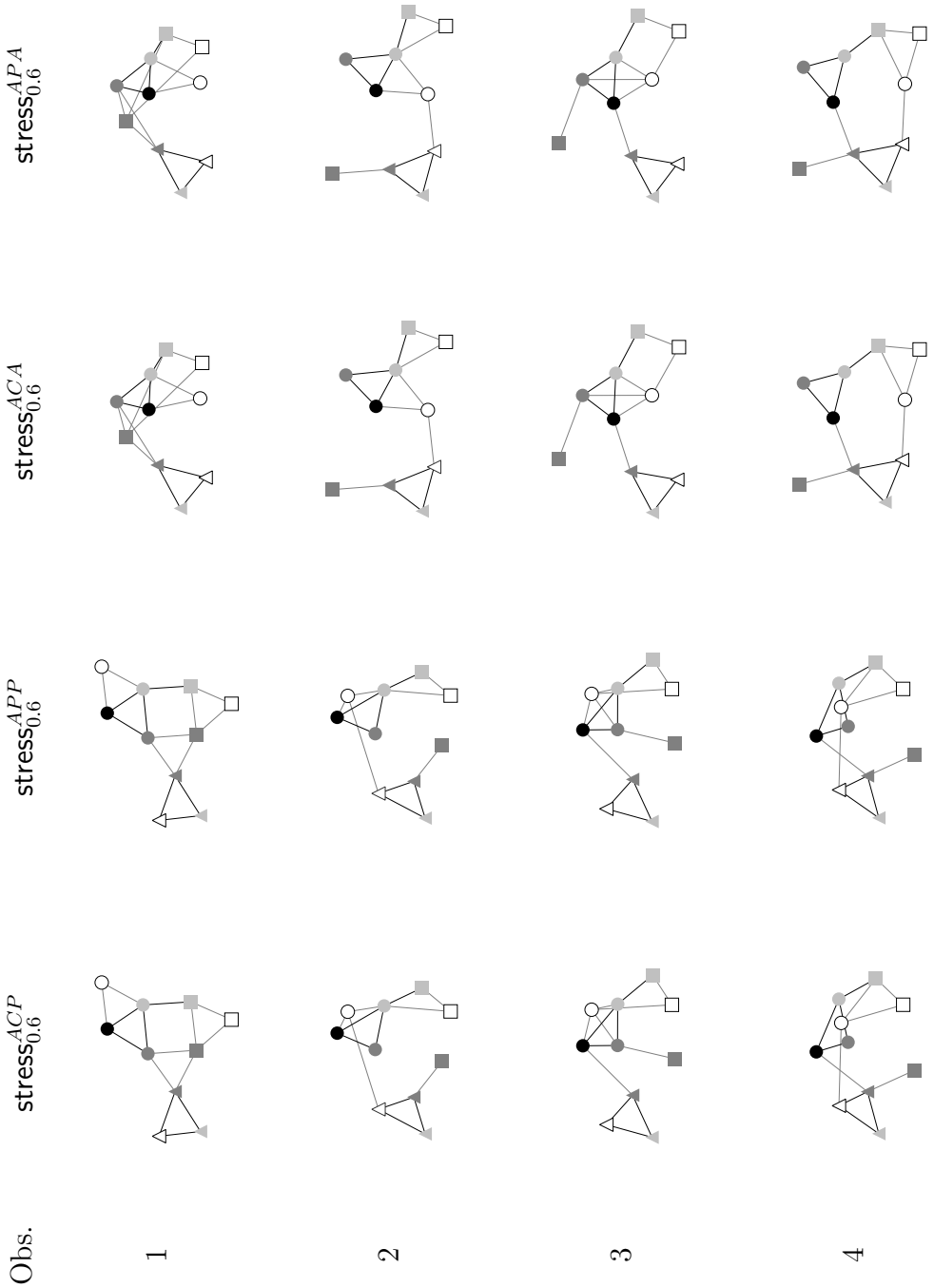


Figure 3.6.: Qualitative Example. Layout by strong anchoring. Initialization has no effect on results, but quality of layouts from anchoring to previous is severely suffering due to bias to earlier observations.

Linked stress In previous work following this approach, vertices where mostly linked only with their direct predecessors and successors. We generalize this to the complete sequence, and formulate a corresponding stress function,

$$\begin{aligned}\mathsf{stress}^L_\alpha\left(P^{(1)},\ldots,P^{(T)}\right) = \sum_{t=1}^{T} \quad & (1-\alpha)\cdot \overbrace{\mathsf{stress}\left(P^{(t)}\right)}^{\text{quality}} \\ & +\alpha\cdot \underbrace{\sum_{i\in V}\sum_{\substack{t'=1\\ t'\neq t}}^{T} \phi_i^{(t)}\zeta(t,t')\left\|p_i^{(t)}-p_i^{(t')}\right\|^2}_{\text{stability}},\end{aligned} \tag{3.3}$$

where $\zeta(t,t')$ is a function controlling the influence of the position at a certain time t for vertices at other time points t'. Concretely, we implemented two version w.r.t. $\zeta(t,t')$:

- $\zeta_G(t,t') = e^{-\frac{1}{2}(t'-t)^2}$, a Gaussian function with mean value t and variance 1 without normalization, i.e., $\zeta_G(t,t) = 1$. Thus, all instances of one vertex are linked together, however, with diminishing influence the further they are apart from each other in time.
- $\zeta_W(t,t') = 1$ for $|t-t'| = 1$, and $\zeta_W(t,t') = 0$ otherwise, i.e., a vertex is only linked within a time-window of size 1. This corresponds to the linking approach most often used in previous work.

Again, parameter $\phi_i^{(t)}$ potentially allows controlling of individual stability per vertex; however, we here only consider $\phi_i^{(t)} = 1$. In the same way as for our anchoring formulation, parameter α explicitly controls the trade off between quality and stability. At its extremes, setting $\alpha = 0$ corresponds to independent static stress minimization of each observation, while $\alpha = 1$ will produce maximally stable layouts, i.e., constant positions for each vertex in each observation.

Variants Regarding initialization, the two options considered here are to use an aggregate layout obtained by minimizing stress^{AGG}, or to use individual classical scaling solutions for each individual observation. Initializing with an aggregate layout corresponds to the simultaneous variant in the framework of Diehl and Görg (2002). In this case, stress of the stability terms is initially zero, and, by minimization of stress^L_α, we expect to improve structures qualitatively for which aggregate positions are inappropriate. On the other hand, initialization by classical scaling should deliver more appropriate solutions with respect to quality terms, and minimization of stress^L_α is expected to subsequently align positions of vertex instance with each other.

Depending on initialization and $\zeta(t,t')$, we obtain four linking methods:

- **LCG** initialization by classical scaling, use ζ_G.

- **LCW** initialization by classical scaling, use ζ_W.
- **LAG** initialization by the aggregate layout stress^{AGG}, use ζ_G.
- **LAW** initialization by the aggregate layout, use ζ_W.

Computation Since the stability model is expressed as zero-distance stress terms like in the anchoring formulation, standard methods for stress minimization can be used in unaltered form. Like in the previous section, all layouts in the sequence are processed by Procrustes analysis after initialization to avoid translational and rotational artifacts, and again after stress minimization.

Using localized stress minimization (cf. Algorithm 1), the sum of votes for each vertex has $T-1$ additional terms compared to static stress when using ζ_G, thus, minimization runs in $\mathcal{O}(T^2n^2)$ time per iteration. For ζ_W, this reduces to $\mathcal{O}(Tn^2)$, since there is only two additional terms for each vertex. Again, computation of the aggregate layout, classical scaling solutions, copying initial positions, and performing Procrustes analysis do not affect overall complexity. Since in most practical scenarios, the number of observations T is small compared to the graph size, total asymptotic complexity is still dominated by computation of shortest path distances.

Discussion Figure 3.7 shows layouts of the small example obtained by linking variants with moderate trade-off parameter $\alpha = 0.2$. In the first two columns, layout calculation is initialized with CMDS solutions, while in the second two columns, the aggregate layout is used.

Regarding initialization with CMDS, the layouts obtained still have a strong resemblance to the static layouts which is especially visible in Observation 1. However, due to integration of offline information, the triangular group is arranged differently, such that the dark square is positioned in the upper part of the layout in Observation 2, which, like anchoring to the aggregate, yields much less vertical movement of the whole structure from Observation 3 to Observation 4, compared to the static layout. We furthermore observe – as an effect of initialization by CMDS and only moderate stability – that the triangular groups still flips in Observation 3 and 4 and that the circular group flips in Observation 2 for both Gaussian and windowed influence functions. Moreover, with Gaussian influence, the same artifact of flipping the light-gray and white circular and square vertices in Observation 3 appears as in offline anchoring with initialization by CMDS, again due to equal positions in the CMDS layout. The reason that this artifact is avoided with the windowed influence function is that, for this short sequence the sum of stability weights is actually lower for Gaussian influence[4], and thus windowed influence actually yields slightly stronger stability at the same value of α.

[4] $\zeta_G(t, t \pm 1) = 0.61, \zeta_G(t, t \pm 2) = 0.14$, while $\zeta_W(t, t \pm 1) = 1$.

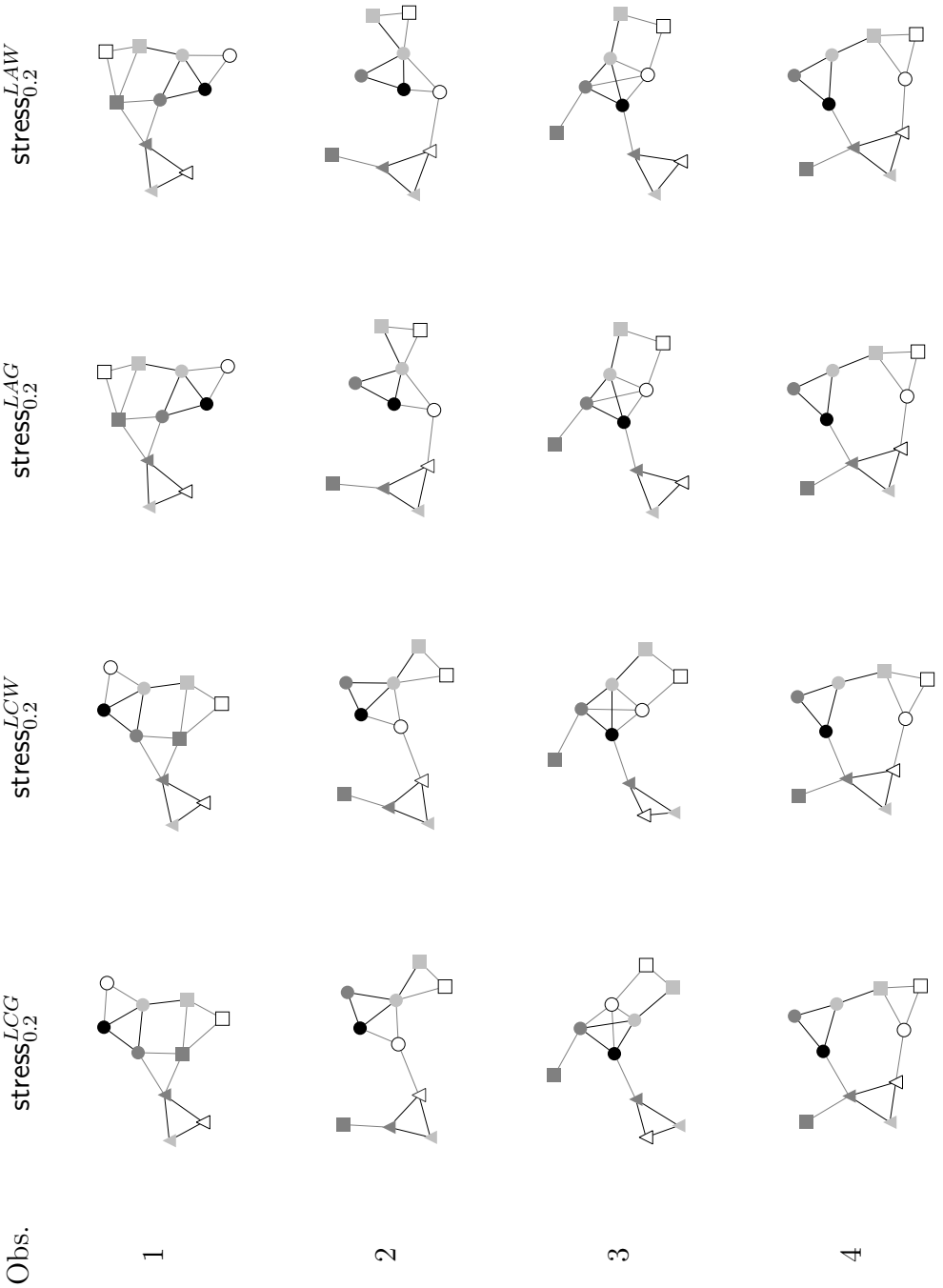


Figure 3.7.: Qualitative Example. Layout by moderate linking. All methods provide readable layouts and little movement, but initialization by aggregate also removes all flips in the combinatorial embedding.

Regarding initialization with the aggregate layout, there is no visible difference between the choices of the influence function. Stability seems slightly higher for the windowed function (for example, the white square moves slightly less in Observation 2), due to the same argument just mentioned. However, initialization by aggregate is very beneficial in terms of flips: there are none. This is because of the arrangement of Observation 1, that differs from all previously considered approaches in positioning the square group above the circular group.

Overall, all linking variants are producing readable representations of each individual structure, but initializing with the aggregate layout provides the least flips and movement.

We next consider linking with strong trade-off parameter $\alpha = 0.6$. The resulting layouts are shown in Figure 3.8. All results are virtually equal for each observation, except for the first one, which, however, exhibits the same principal arrangement. There are slight differences in placing the circular group vertically, but there is no apparent explanation for this behavior by the different principles of our variants.

Similarly to offline anchoring, quality is only severely affected in the (first) outlier observation, while movement is largely reduced.

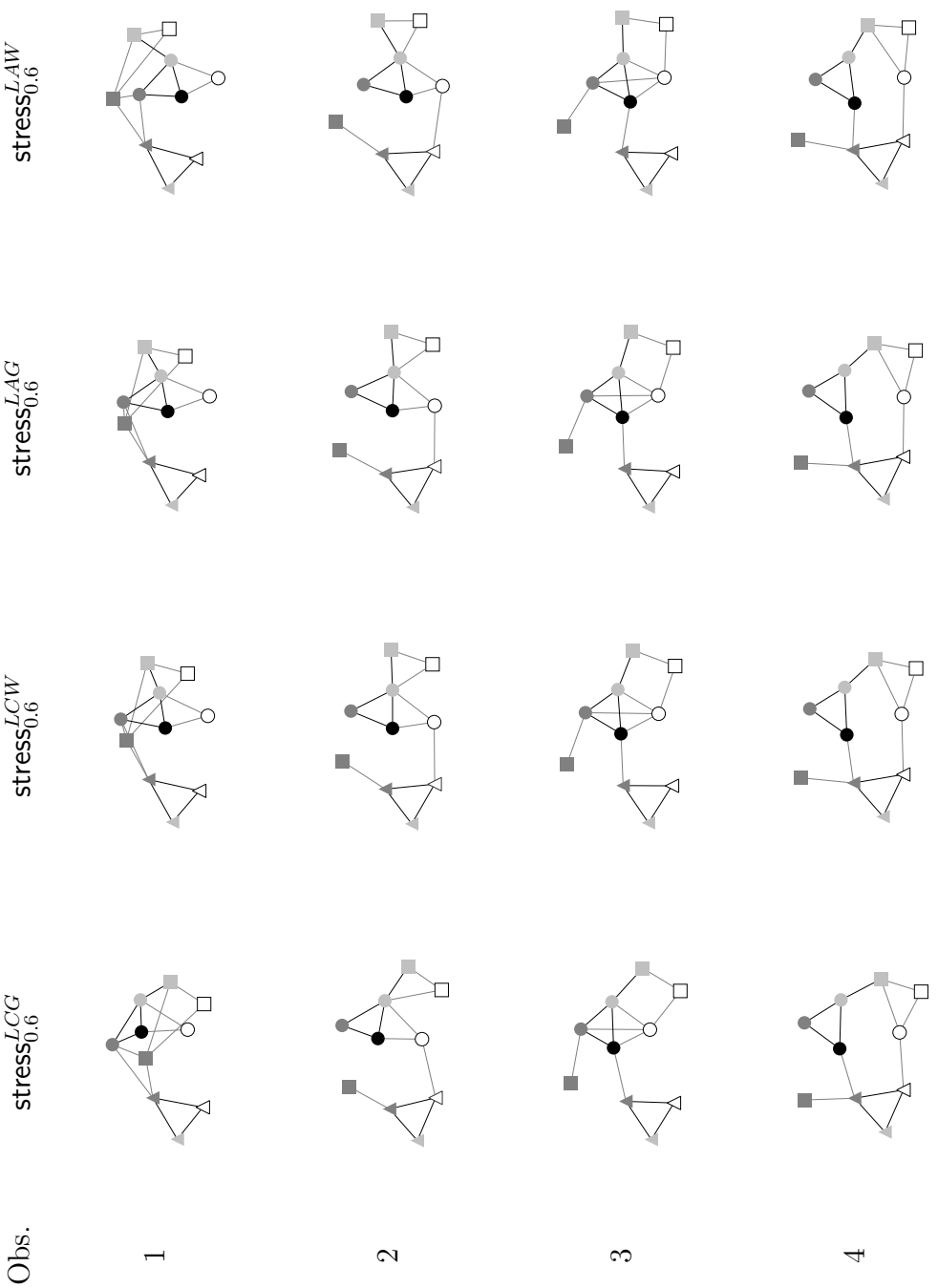


Figure 3.8.: Qualitative Example. Layout by strong linking. All variants result in essentially the same layout, largely reducing movement.

3.3. A larger example

The discussion of variant approaches by the small illustrative example used in the previous section gave insight to their distinct properties. In the following, we apply our methods to a real world data set to see if these still hold for larger and more complex data, and to demonstrate practical effectiveness in a realistic scenario.

Concretely, we use a longitudinal network of acquaintanceship among university students. The data is courtesy of Britta Renner and Manja Vollmann (Department of Psychology, University of Konstanz) and was collected in 15 waves between October 2008 and February 2009.

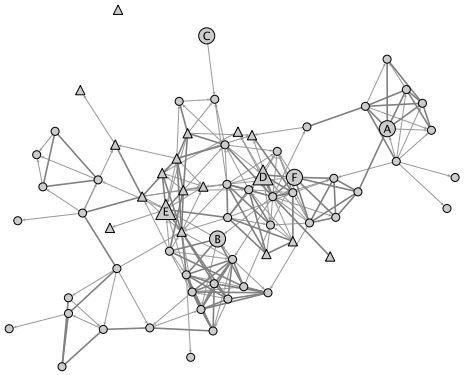
Students provided, among other data, their current perceived level of acquaintanceship with each other on a scale from 1 (lowest) to 7 (highest). We dichotomized each observation using 5 as a threshold. Of the 78 freshmen majoring in Psychology, only nine did not participate in an initial screening, never answered any questionnaire, or never made a nomination resp. were never nominated at a level above the threshold. The example networks thus consist of acquaintanceship nominations among 69 actors (18 male, 51 female) that form a connected component when aggregated over all waves.

All layouts shown in the following have been computed with all 15 observations as input; for compactness, we here only show layouts for Observation 10 to Observation 14. The full sequence is depicted in Brandes, Indlekofer, and Mader (2012). In all figures, female actors are represented by circles, and male actors by triangles. If an edge is reciprocated, i.e., two students mutually nominated each other, it is represented as a thicker line segment; otherwise, the edge is represented as a thinner arrow.

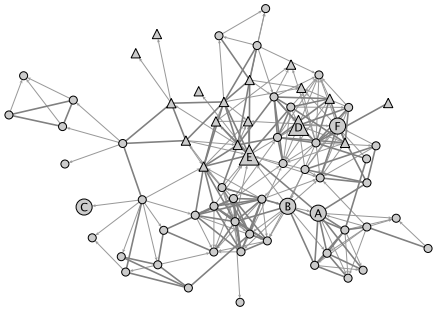
Static stress Figure 3.9 shows layouts of Observations 10–14 as obtained by static stress minimization using classical scaling as initialization. Subsequent layouts are aligned to each other by Procrustes analysis. Besides sparse peripheral structures in the left-hand part of each layout, we can observe four major relatively stable groups: Two female groups, one around Actor A, and another one left of or below Actor B, a male group around Actor E and a mixed group involving Actors D and F.

Aggregation Global layouts with fixed positions for each vertex are given in Figure 3.10, using our proposed method of stress applied to aggregated distances (stress^{AGG}) on the left-hand side, and using static stress applied on the supergraph ($\mathsf{stress}^{\bar{G}}$) on the right-hand side. Figure 3.11 juxtaposes layouts of the individual observations inheriting the respective global layout.

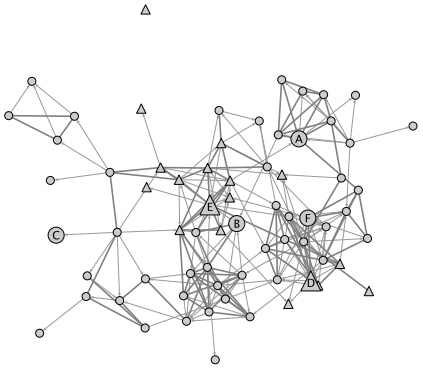
Since there is an abundance of short distances in the supergraph, the global layout obtained by $\mathsf{stress}^{\bar{G}}$ appears more clumped together, and thus, cluttered. Even if scaled to take the same area as the global layout obtained by stress^{AGG}, treating both momentary and stable connections equally distorts differentiation of stable structures. For example,



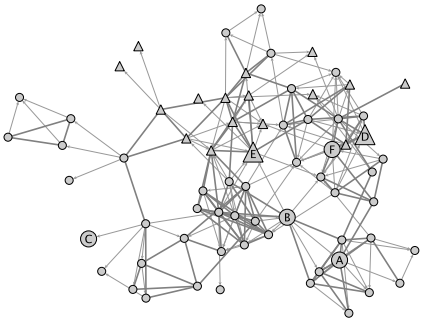
(a) Observation 10



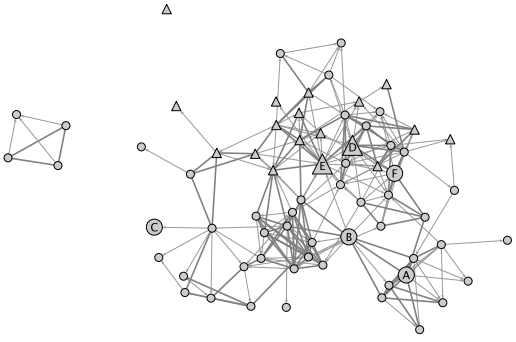
(b) Observation 12



(c) Observation 11



(d) Observation 13



(e) Observation 14

Figure 3.9.: Observations 10–14 of the acquaintanceship network drawn by static stress minimization. Circles represent female actors, whereas triangles represent males. If an acquaintanceship tie is reciprocated, it is represented by a thicker line segment, otherwise an arrow indicates the direction of nomination. Some vertices are indexed for later reference.

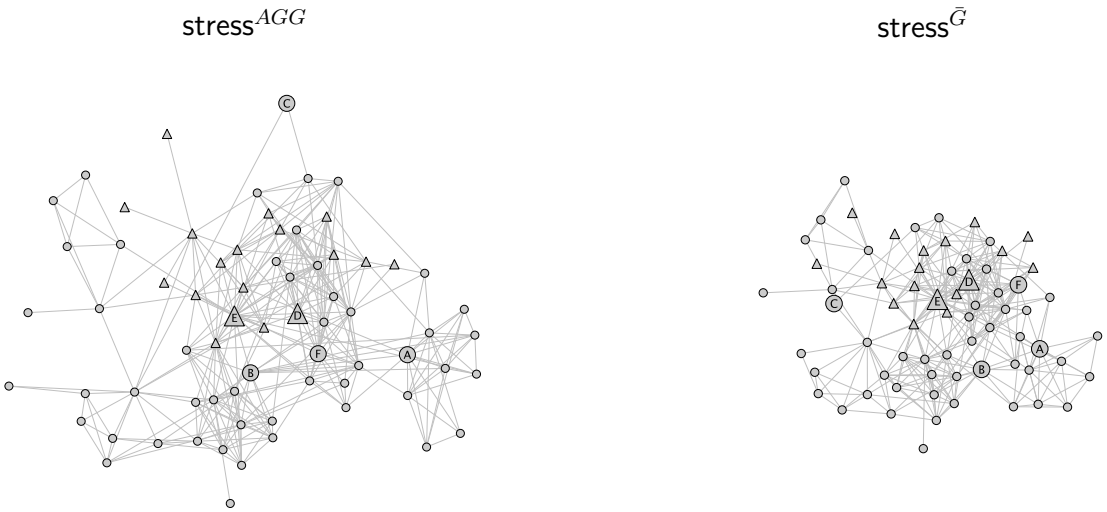


Figure 3.10.: Global layout of the acquaintanceship network using aggregate layout (left) and static stress minimization of the supergraph $\bar{G}$ (right). An edge is drawn if two vertices are adjacent in any graph of the input sequence.

it is hard to recognize that the male group around Actor E is only loosely connected to the mixed group of Actors D and F until Observation 12. Although less organic, the layout obtained by stress^{AGG} more faithfully captures the global structure of the network, allowing to better discern the major stable substructures mentioned above.

Due to perfect stability, occurrences of the same actor are easily identified in each of the sub-figures of Figure 3.11. However, a comparison with the layouts obtained by regular stress minimization shown in Figure 3.9 reveals that this extreme stability comes at the price of less desirable individual layouts:

- Sub-configurations may appear to be placed oddly at times. An example is the group around Actor A. Using stress$^{\bar{G}}$, it is placed geometrically close to the group of Actor B, although they are actually very distant up to Observation 11 (cf. Figure 3.9a). On the contrary, using stress^{AGG} it would be expected to lie closer to the group to which it connects through Actor B in Observations 11–14. Here, its position leads to unnecessary long links representing the strong ties to Actor B that have been established by then. These also create much clutter in the group belonging to Actor F.

- The almost bipartite structure of the group around Actor A that could be observed in Figure 3.9 in Observation 13 and 14 is hardly recognizable.

- Actor C starts out being connected to an actor at the very top, but then links up with an actor further down. Since this new neighbor is far away from the first neighbor, any reasonable constant position can only result in either an awkward or a long link connecting Actor C.

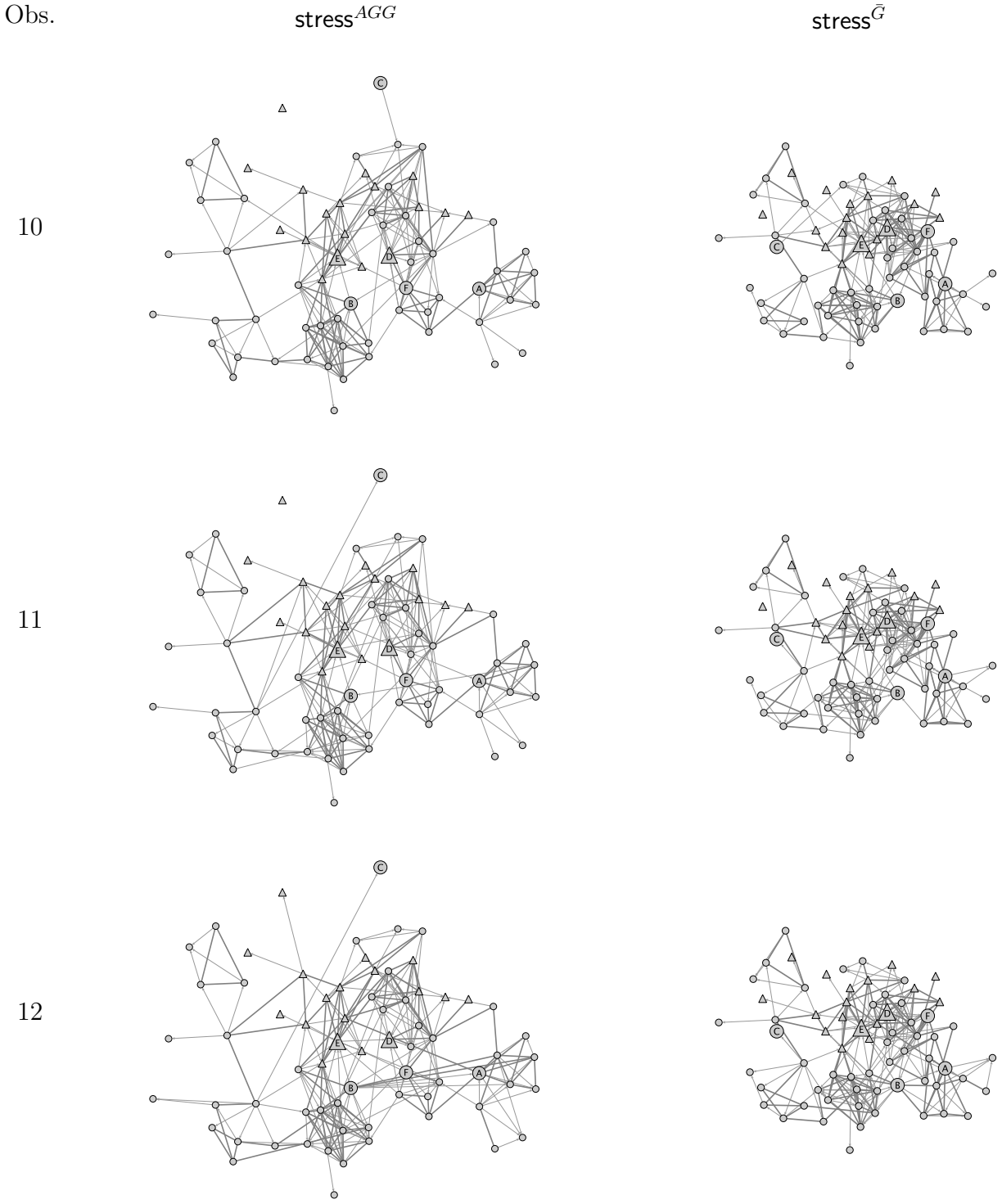


Figure 3.11.: Observations 9–12 of the acquaintanceship network using aggregate layout (left) and static stress minimization of the supergraph $\bar{G}$ (right). Observations 12–14 on the next page.

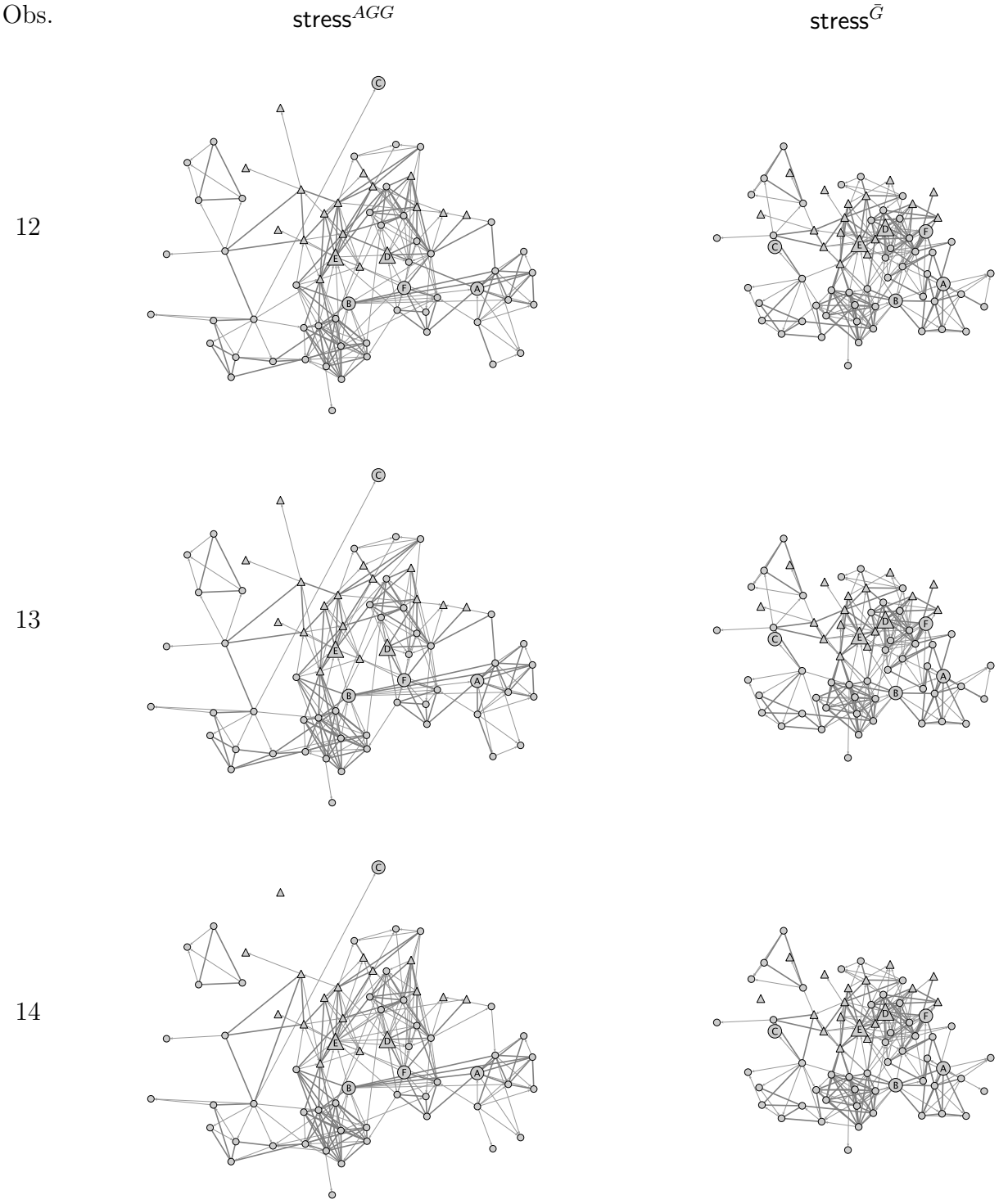


Figure 3.11.: Observations 12–14 (continued from previous page). Treating all connections equally, layouts obtained by stress$^{\bar{G}}$ appear more cluttered. Stable substructures are more easily discernible using stress^{AGG}.

- Actor D and Actor E exhibit a very volatile connection, resulting in corresponding changes of geometric distances between them in Figure 3.9. Although this volatility can be observed in the aggregate layout, it might be easily missed if the actors were located in denser regions.

These qualitative deficits of aggregate layouts are corroborated by quantitative measurement; for instance, stress of the layout of Observation 13 and Observation 14 as measured by Equation 2.2 is almost thrice as high as in the corresponding layouts of Figure 3.9.

The above examples nevertheless illustrate that fixed positions are helpful in building a mental map of the overall configuration, but that it is desirable to allow for at least small deviations to represent better the specific configurations at individual time points.

Anchoring Our sample observations are shown in Figure 3.12, now laid out using moderate anchoring ($\alpha = 0.2$) with reference positions taken from the aggregate layout of Figure 3.10 (left-hand side). Although more effort is needed to trace vertices from one layout to the next, major substructures remain stable and the individual quality of layouts has improved notably.

This is confirmed when considering the issues we had with the previous aggregate layouts. The groups of Actors A and B move towards each other when they start being connected (Observations 11–14). The near-bipartite structure of the group involving Actor A is apparent in Observation 14. Movement highlights the changing affiliations of Actors C and the volatile connection between Actor D and Actor E. While it is more difficult to locate them in individual layouts, deviations from their reference position are actually meaningful and therefore considered desirable.

Next, we want to demonstrate the benefits of using offline information. Compare the above results of anchoring at reference positions from an aggregation layout (Figure 3.13, right-hand side) to an online approach based on initialization with the preceding layout and no control for stability (Figure 3.13, left-hand side). Note that the latter is similar to the online approach provided by the SoNIA[5] software tool (Bender-deMoll and McFarland, 2006), since the method employed there is conceptually the same, as is the basic objective function, stress, when using the Kamada-Kawai option (MultiCompKK).

Both approaches result in a similar configuration for Observation 10. In Observation 11, the group of Actor A establishes lasting connections to the groups of Actor B and momentary connections to the group of Actor E, forcing the groups to move towards each other. The online approach results in a positioning of the group of Actor A above the group of Actor F in Observations 11 and 12. However, with the dissolution of the momentary connections in Observation 13, those groups are forced to flip around each other, creating excessive movement between observations.

[5] `http://www.stanford.edu/group/sonia/`

Anchoring with reference positions from an aggregation layout exploits offline information to avoid the need for such flips, and thus eliminates some movement that indicates structural changes inappropriately. Observe that there is much less difference in the configuration of substructures than suggested by the large movements in the online scenario.

Linking Finally, Figure 3.14 juxtaposes our variant offline methods, namely anchoring to aggregate positions (stress^{APA}, right-hand side) and linking (stress^{LAG}, left-hand side), both with moderate control for stability ($\alpha = 0.2$). The relative positioning of larger structures, and even of most individual vertices, is nearly identical. Hence, linking also avoids large flips and excessive movement. However, compared to offline anchoring, we observe an improvement in qualitative detail, especially with respect to representation of the group around Actor A. The latter is better separated from the group of Actors D and F when it moves closer to the groups of Actors E and B in Observation 11. Furthermore, the individual structure of this group is represented more clearly in Observations 12–14, such that the layout of Observation 13 now also reveals the group's near bipartite structure.

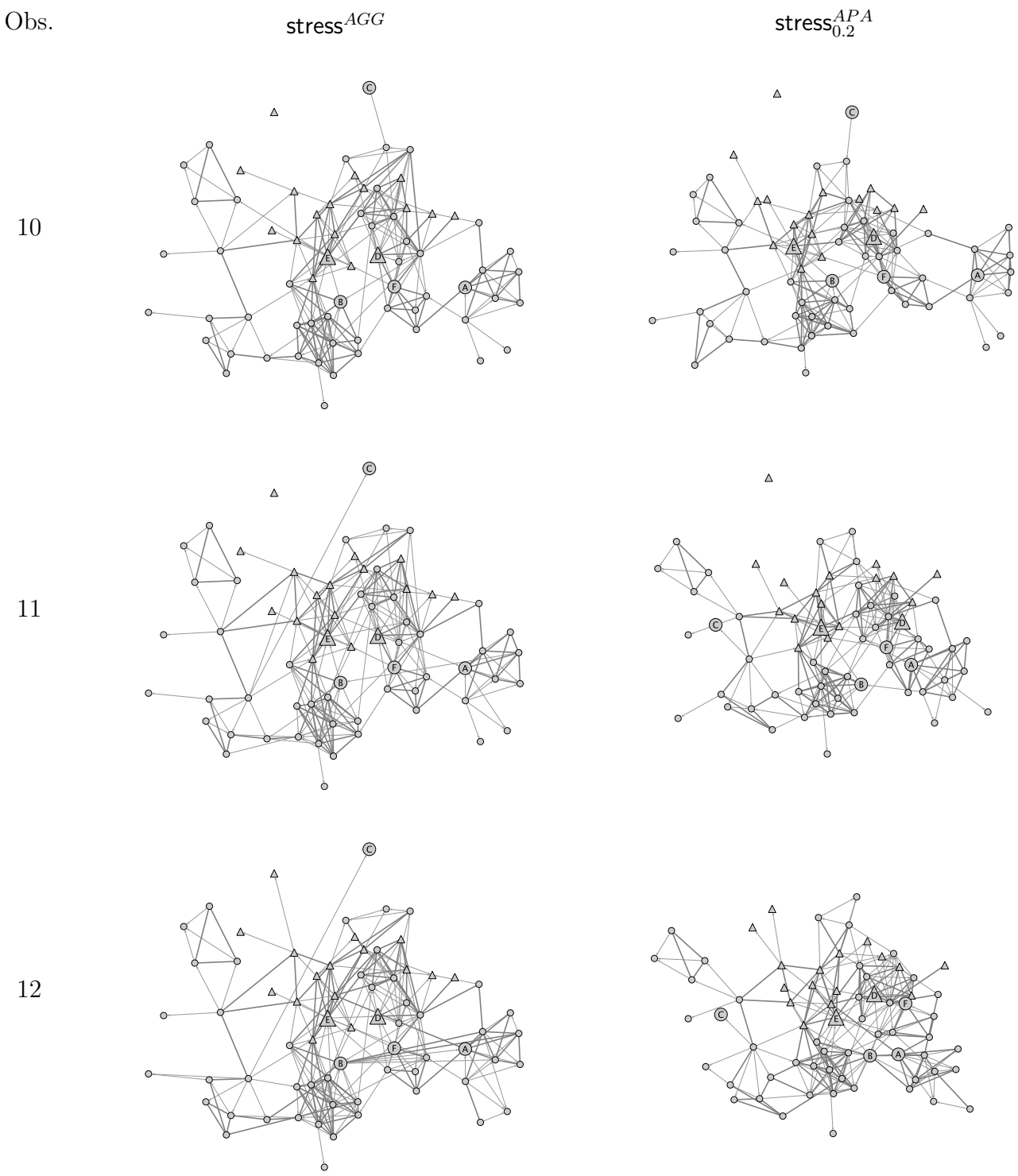


Figure 3.12.: Observations 9–12 of the acquaintanceship network using aggregate layout (left) and moderate anchoring with aggregate as reference (right, $\alpha = 0.2$). Observations 12–14 on the next page.

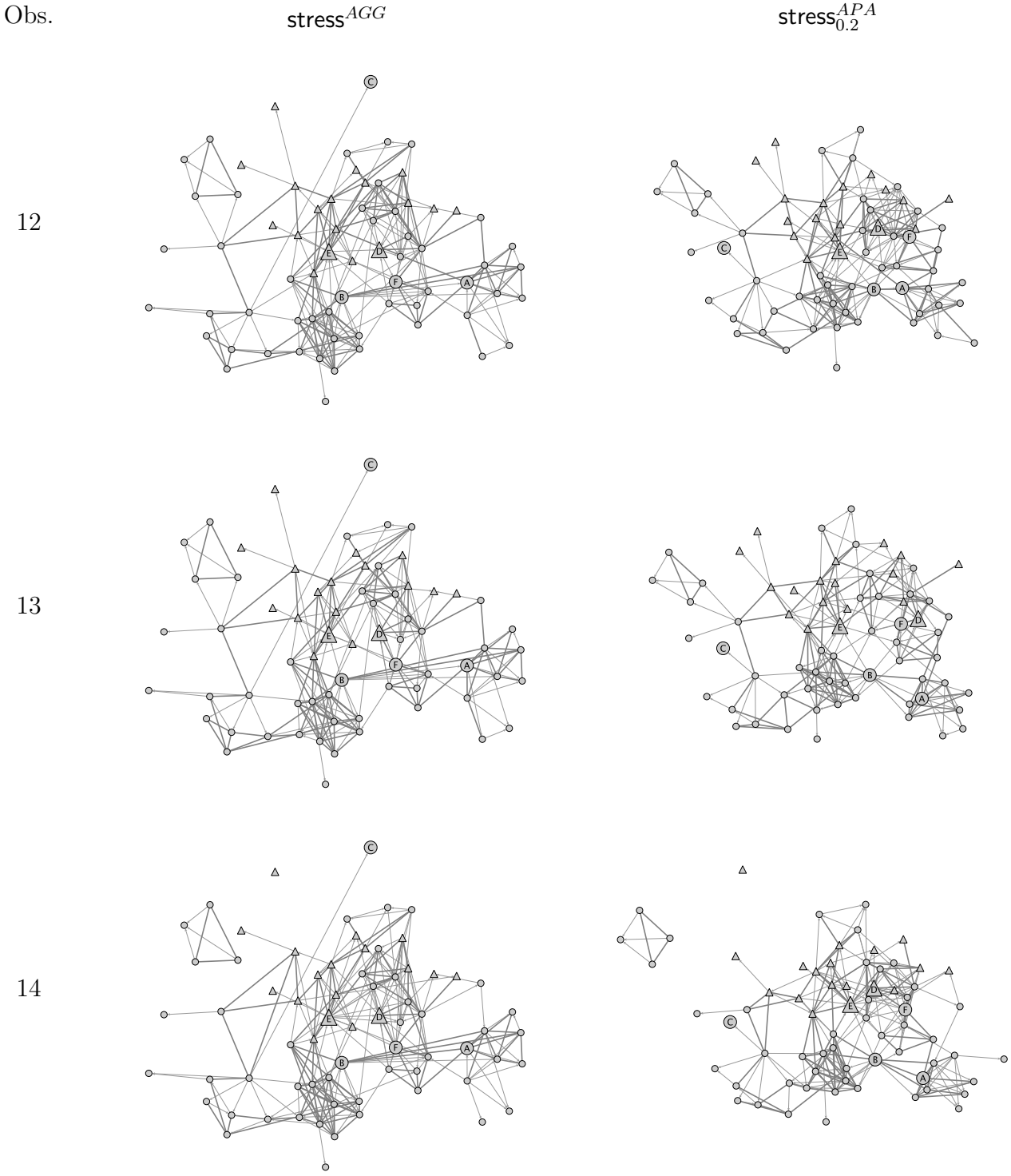


Figure 3.12.: Observations 12–14 (continued from previous page). While movement makes it harder to trace individual vertices, it also remedies inherent drawbacks of the aggregate layout.

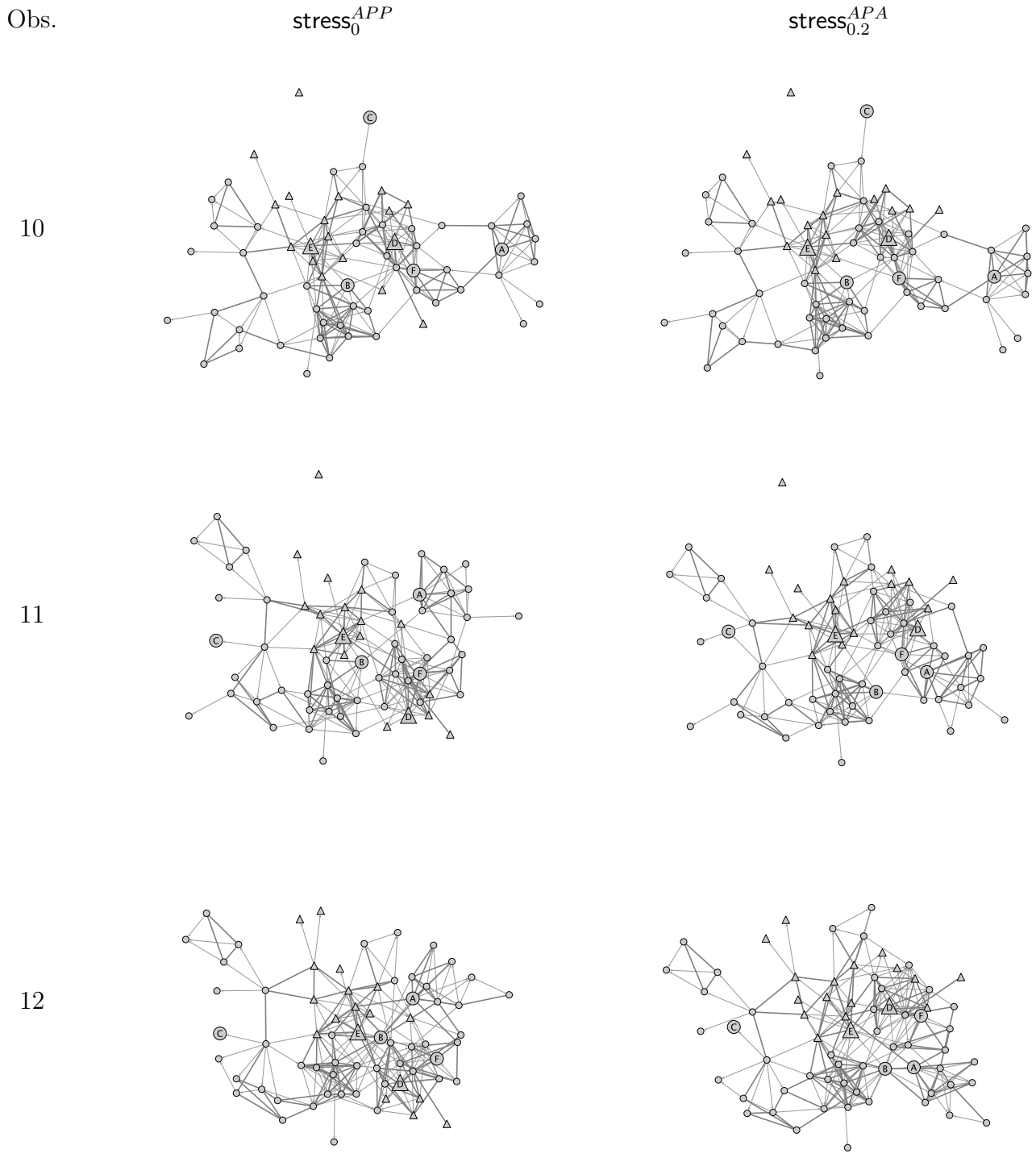


Figure 3.13.: Observations 10–12 of the acquaintanceship network drawn by a common online approach (left, static stress with initialization by previous) and by moderate anchoring with aggregate as reference (right, $\alpha = 0.2$). Observations 12–14 on the next page.

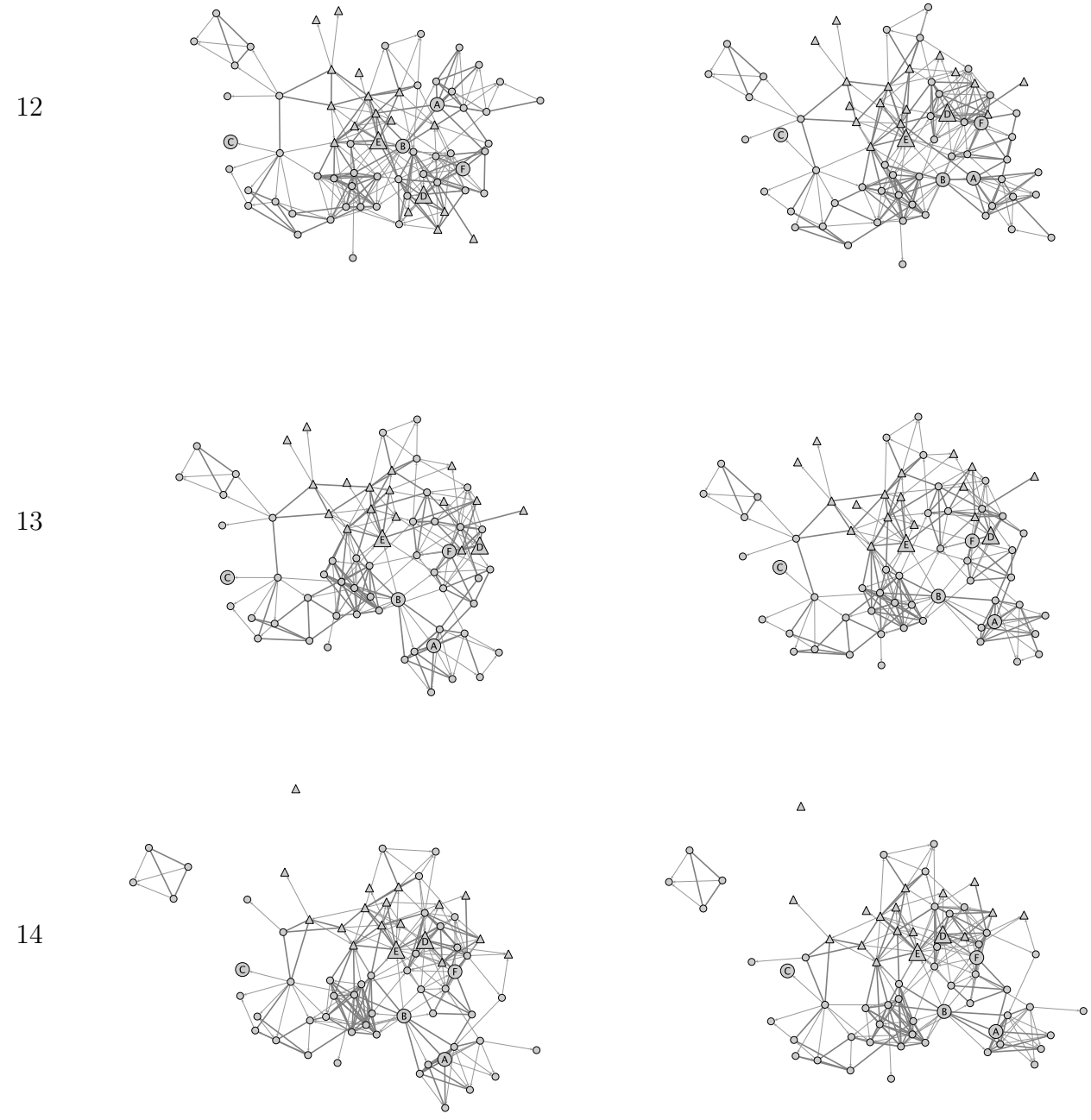


Figure 3.13.: Observations 12–14 (continued from previous page). Substructures are similar, but the offline method avoids the need for flipping subgroups such as the ones associated with Actors A and F from Observation 12 to 13.

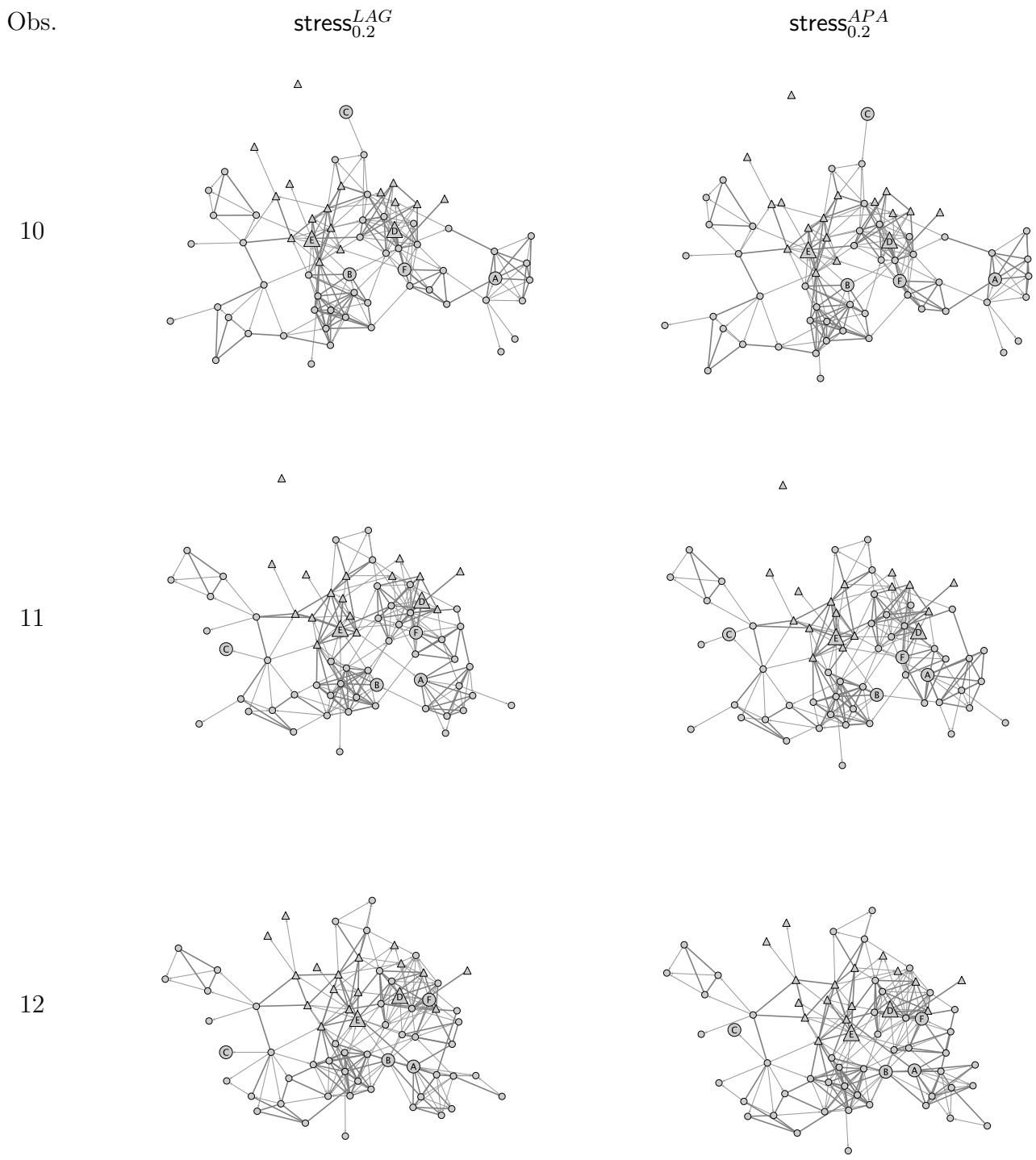
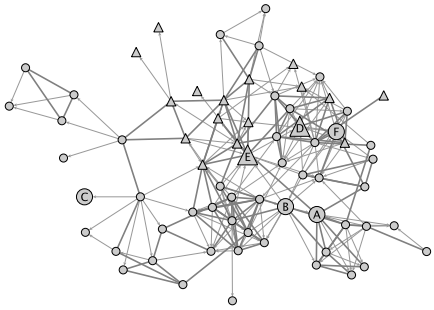
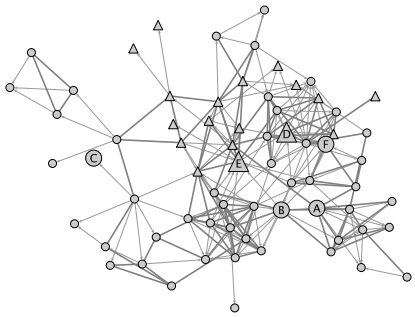


Figure 3.14.: Observations 10–12 of the acquaintanceship network drawn by moderate linking (left, $\alpha = 0.2$) and by moderate anchoring with aggregate as reference (right, $\alpha = 0.2$). Observations 12–14 on the next page.

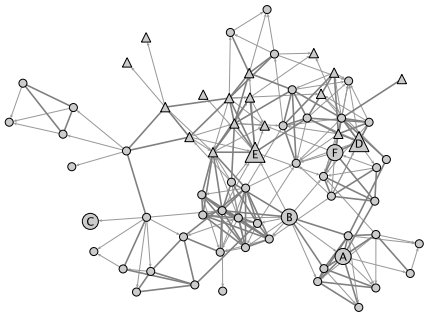
Obs. $\mathsf{stress}_{0.2}^{LAG}$ $\mathsf{stress}_{0.2}^{APA}$

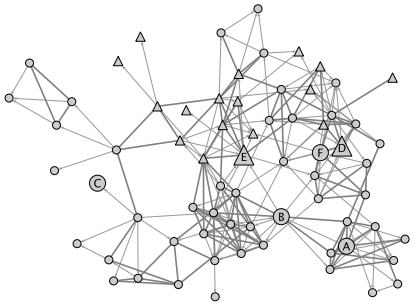
12



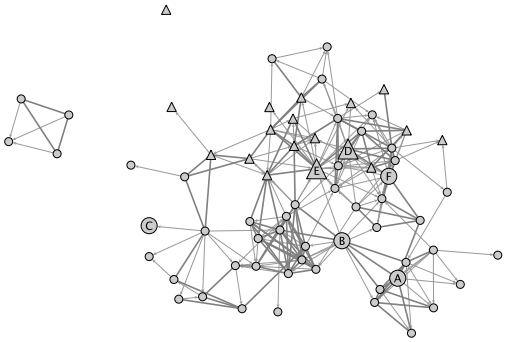


13





14



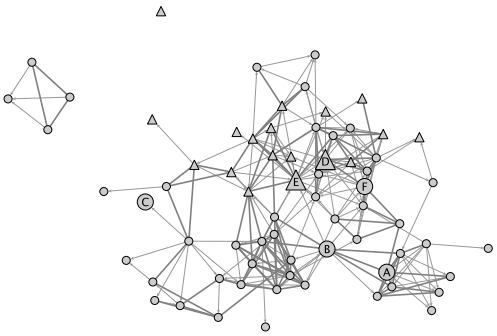


Figure 3.14.: Observations 12–14 (continued from previous page). Arrangements of larger structures in individual observations are almost identical, but the linking approach better represents smaller substructures like the group around Actor A.

4. Experimental evaluation

> *"To understand information processes, computer scientists must observe phenomena, formulate explanations, and test them. This* is *the scientific method."*
>
> *(Tichy, 1998)*

How can an algorithm such as stress minimization be evaluated? Its heuristic nature renders theoretical assessment difficult, if not impossible. One can, of course, investigate the algorithm's behavior by observing its effect on example data, as we have done in the previous chapter. This is a most natural process for the development of any algorithm, and indeed, for many types of general research. However, merely illustrative evaluation obviously can claim neither certainty nor completeness.

Typical experimentation in algorithmics usually deals with assessing runtime and space complexity in application typical environments. Here, the case is different: We are interested in qualitative behavior of our visulization algorithm. To be more precise, the goal is to assess the ability of dynamic stress minimization to trade-off individual layout quality with dynamic stability, depending on input characteristics, the different variants proposed in the last chapter, and algorithm parameters.

We can draw on several disciplines to be able to perform this kind of assessment: Experimental algorithmics, algorithm engineering, and design of experiments. To be able to employ a hypotheses-based experimental design following those disciplines, we provide a way to generate random, yet application-typical, instances of dynamic networks, and develop sound measurement of the trade-off behavior of our algorithms.

4.1. Experiments on algorithms

Before we delve into evaluating the stress minimization approach for dynamic graph drawing, we give a brief overview of experimentation in theoretical computer science, and methodology related to experimental analysis of algorithms, a field that is generally referred to *experimental algorithmics* (Moret, 2002).

Experimental analysis & hypotheses When we speak about algorithm analysis, we immediately think of abstraction of algorithmic ideas to pseudo-code and computer models, analysis of asymptotic complexity, and mathematical proof. However, there are many reasons, why experimental analysis of algorithms might be appropriate. The reason most important and relevant to our case is, that theoretical characterizations of strengths and weaknesses of an algorithm are hard or even impossible to come by. David S. Johnson, one of the initiators of experimental algorithmics, actually refers to this type specifically as *experimental analysis*, next to three other types of experimental studies (Johnson, 2002).

In the same article, Johnson derives a list of principles that should guide experimental analysis of algorithms, along with pitfalls and pet peeves the experimenter should be aware of, and suggestions to avoid the latter. While it is surely worthy to follow these principles in any experimental endeavor, Johnson's guide does not incorporate a crucial aspect of the traditional scientific method: testing theories by experimental testing of hypotheses.

The Oxford English Dictionary[1] states that the term *scientific method* "is now commonly represented as ideally comprising some or all of systematic observation, measurement, and experimentation, induction and the formulation of hypotheses, the making of deductions from the hypotheses, the experimental testing of the deductions, and (if necessary) the modification of the hypotheses".

Initially, a hypothesis often is an educated guess on the basis of prior knowledge or observation. More formally, it should contain a tentative explanation of the process under consideration, and might include prediction of the outcome of an experiment. The hypothesis usually relates two or more variables that are then used in a controlled experiment to test the hypothesis: the independent variable(s) that the experimenter controls for, and the dependent variable(s) that informs about the effects under study. To count as a *scientific* hypothesis, it must be *testable* or *falsifiable*, meaning that there must be the possibility that counterexamples to the hypothesis exist, and that it is feasable to conceive those practically as outcomes of experiments (Popper, 1934). While the outcome of an experiment cannot prove any hypothesis, it can support a hypothesis, if and only if the latter is testable.

Algorithm engineering In the late 90's, the discipline of algorithm engineering emerged with the goal to reconcile classical algorithm theory and experimental algorithmics. It provides a main cycle consisting of design, analysis, implementation, and experimentation (see Figure 4.1) as a guide to research and development of algorithms (Sanders, 2009; Chimani and Klein, 2010). At the center of the cycle are falsifiable hypotheses arising from design, analysis, implementation, or previous experiments, which are then evaluated by experiments. The result of the experiment can then support or falsify the hypotheses, or lead to refined ones. The main goals are to supplement theoretical

[1] `www.oed.com`

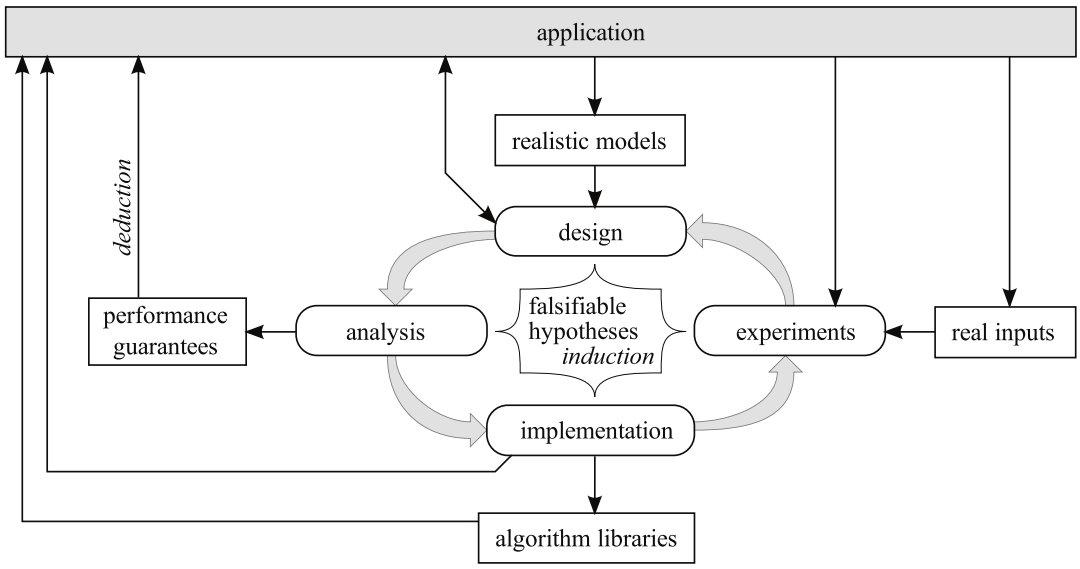


Figure 4.1.: The algorithm engineering cycle (Chimani and Klein, 2010).

analysis, and importantly here, provide deeper insight into algorithmic behavior. Other goals less relevant to us are related to deployment of algorithms to applications in form of tested and robust implementations collected in algorithm libraries.

Design of experiments While the algorithm engineering cycle provides a basis for general research and design of algorithms, it is still unclear how the experimental apparatus might look like. To get a better understanding, we can draw from a field generally referred to as *design of experiments* originating back to Fisher (1935). The field mostly covers statistical methods dealing with variation that can occur in an experiment. Here, we are more interested in general principles and terminology related to experimental design, and the implications for our algorithmic experiments. A general overview for design of experiments in the traditional sense can be found in the introductory chapter of Montgomery (2013), while Rardin and Uzsoy (2001) provide a good starting point regarding aspects of experimental design relevant to optimization algorithms.

A designed experiment is usually conducted to test a hypothesis about certain effects of variables the experimenter controls for on the outcome of the process under study. The independent variables are often referred to as *factors*, and the dependent variables as *response*. A basic representation of a designed experiment is thus:

$$\text{factors} \quad \rightarrow \quad \text{experiment} \quad \rightarrow \quad \text{response}$$

Table 4.1 provides a procedural breakdown for conducting designed experiments by Montgomery (2013). In the following, we will elaborate on the topics of factors and responses, and on experimental design.

1.	Recognition of and statement of the problem
2.	Choice of factors and levels
3.	Selection of a response variable
4.	Choice of experimental design
5.	Performing the experiment
6.	Data analysis
7.	Conclusions and recommendations

Table 4.1.: Procedure of a designed experiment recommended by Montgomery (2013).

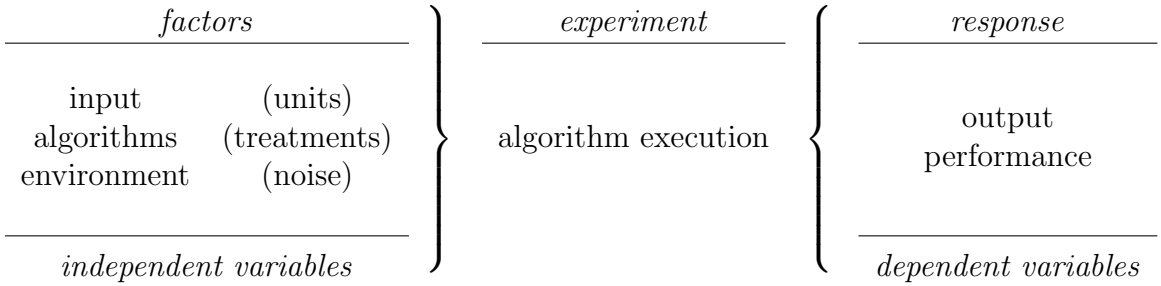


Figure 4.2.: Algorithm execution treated as experiment.

Generally, factors can be divided into *experimental units*, which are the objects that the experiment is performed on, and into *treatments*, which usually correspond to variations of the parameters of the examined process. For example, in a classic medical experiment to test effectiveness of some drug, the units could be humans or animals (or groups thereof) and treatments could be different chemical variations of the substance.

How can this be translated to empirically evaluate algorithms? The very constant of any algorithm is that it retrieves some input data, processes this input, and eventually yields output data:

$$\text{input} \quad \rightarrow \quad \text{algorithm} \quad \rightarrow \quad \text{output}$$

Observing the similarity of these representations, algorithms themselves can be seen as an experiment. However, this picture must be refined slightly, since usually the algorithmic experimenter often must compare several algorithms, or variants of one algorithm, as is the case here. Thus, it is more natural to consider algorithm variants as different treatments to different inputs, which in turn are then considered the units of the experiment. The experiment itself then merely corresponds to algorithm execution. Regarding the response, experimental algorithmics is most likely interested in either qualitative aspects of the solution, or in performance characteristics like runtime and space consumption. Figure 4.2 summarizes this view.

In our case, the experimental units are different kinds of sequences of dynamic graphs, while the treatments are the algorithmic variants presented in the previous chapter,

and parameters thereof, that is, choice of initialization and reference layout, and trade-off parameter α. Thus, the various factors are quantitative and qualitative ones. For the former, the experimenter has to select levels of factors either by choosing specific values, or by sampling at random. Depending on which measurements are performed, the problem of selecting appropriate levels applies to the response as well.

Having set factors and responses of the experiment, the main task is then to fix what is referred to as the *experimental design*, which deals with handling potential experimental error.

An important error that needs to be ruled out is due to *confounding effects*. In the medical drug test example, the result might also depend on the age of the experimental unit, so the experimenter has to control for it. However, in most cases not all confounding effects can be controlled for, or are even known to the experimenter. As a remedy, *randomization* helps to control for systematic effects in the presence of confounding effects or other unknown sources of discrepancy, both by choosing experimental units from the population at random, or by randomly assigning units and treatments, if testing all pairings is too costly.[2]

Another possible error is due to noise, either in the environment of the experiment, or in the measurement of responses. In the medical example, physical differences in the environment of the room that the experiment is performed in, like temperature, might affect the results; or measurement of a response like blood pressure might be erroneous depending on the device used. While measurement error is rarely the case in algorithmic experiments, performance analysis could suffer from noise due to system load or background operations on the experimental platform. Another source of noise emerges if the algorithm performs random decisions during execution. The typical way to deal with noise is *replication*, that is, performing the test on one unit-treatment pairing several times. Both problems do not exist in our case, since the algorithms are deterministic, and measurement of the response will be independent of noise from the environment that the algorithm is run in. However, since we will use random data generators to provide a suitably large input basis, replicated generation of input data with specific settings is needed to control for random differences in generated data.

A third source of error is due to known sources of discrepancy among experimental units. In the case of the drug test example, the drug could behave differently between male and female specimen. In more complex situations, where homogeneous groups of units are conceivable, but where it is too costly to – or not clear how to – separate these by more detailed factors, a typical strategy is *blocking* experiments to these group of units. While contradicting the principle of randomization, blocking separates variability among units from the experimental error, thus increasing precision. The perhaps most widespread strategy is called *randomized complete block design*, in which randomization is applied within blocks, and all treatments are applied to all blocks. This is also the design that we will make use of.

[2] An experimental design testing all pairings of factors is referred to as *full factorial design*.

Experiments in graph drawing While there apparently is a sound methodology available to perform algorithmic experiments in the form of algorithm engineering principles and general mechanisms from design of experiments, it is surprising that only few examples follow those strategies in the field of graph drawing.

Most articles make do with mere illustration of the results of their algorithms on select examples, as, for example in the seminal work of Eades (1984) on the spring embedder. Even works specifically dedicated to experimentally evaluate graph drawing algorithms or to experimentally compare different ones, like Himsolt (1995) and Vismara et al. (2000), are mostly explorative in the sense that hypotheses are only implicitly stated, or – more often – conclusions are only drawn a posteriori.

The only general exception to this observation are user studies in graph drawing, for example, Purchase, Cohen, and James (1997), who analyze the effectiveness and efficiency with which the human can perform different tasks on different layout variants. However, even those kinds of studies are often explorative in nature, for example Purchase (1998). The reason that, largely, a more sound experimental methodology is made use of in those studies is probably due to the close relation to psychology, where rigorous experimentation is a traditional requirement. However, the setting in user studies is different to algorithmic experimentation, since the experimental units are humans, and the treatments are the output of the different algorithms under study. In this sense, algorithmic experimentation is even more needed, to be able to better control treatments in user studies.

Experiments related to dynamic graph drawing Regarding dynamic graph drawing specifically, there exist only very few experimental studies on algorithmic behavior. Again, most experimental evaluation is dedicated to the aspect of human cognition, especially towards the choice of small multiples versus animation for displaying dynamic layouts of sequences, as listed in Section 3.1 from the previous chapter.

One of the few articles dealing with the trade-off behavior between quality and stability in dynamic graph drawing is the user study by Purchase and Samra (2008). Users were faced with graph drawings produced by a linking type algorithm, employing three chosen levels of stability in terms of a trade-off parameter similar to our trade-off parameter α. The experiment revealed that the participant's performance in answering questions regarding structural change was best with low stability level drawings.

Bridgeman and Tamassia (2000) conducted another user study related to dynamic graph drawing. They study how well several measures for similarity between orthogonal drawings of different, but structurally similar graphs reflect a humans perception of similarity between drawings. The measures are categorized into similarity measures of point sets, of positional difference between same instances of a vertex, of change of relative ordering and of change of neighborhoods. A measure for the change of edges is also defined, but this only applies to orthogonal drawings. One result is that similarity measures based on

positional difference are among the best measures. While the results regarding the agreements of these similarity measure and the human perception do not necessarily apply to straight-line drawings, the article still serves as a repository for potential candidate measures for stability between graph layouts.

To our knowledge, the only two articles related to dynamic graph drawing that employ algorithmic experimentation in the sense of treating algorithm execution as experiment are by Lyons, Meijer, and Rappaport (1998) and Frishman and Tal (2008).

As a reminder, the layout adjustment algorithms of Lyons, Meijer, and Rappaport (1998) aim to improve vertex distribution to remove clutter, while maintaining the mental map by an anchoring mechanism. To compare the performance of the two algorithmic variants, they generate random graph layouts with a set number of clusters whose vertices are all placed in a small region. They define a measure for vertex distribution, and two measures for layout stability: the *distances moved* measure is based on the sum of positional differences, while the λ-*matrix* measure is based on relative ordering.[3] Both measures are normalized by an upper bound.

Frishman and Tal (2008) are more closely related to our goal of assessing the ability to trade off individual layout quality and dynamic stability. They demonstrate that their online anchoring force-directed approach still produce "good" layouts while laregely increasing stability by evaluating layout sequences containing two graphs each. The first graph in each sequence is one of five different real-world graphs, while the second one is obtained by randomly changing the first graph's nodes and edges – the authors state that $|E|$ and $|V|$ are modified by up to 15%. Layout quality is measured by the remaining energy, similar to stress, while stability is measured as the average displacement of vertices. This simple experiment works because of two reasons: firstly, measures of the anchoring approach are only evaluated against corresponding layouts by purely static layouts[4], and layout withouth explicit control for stability, that is initializing the second layout with the first one; secondly, measurement is block by the five different input data, thus accounting for their intrinsic differences. However, the experiment does not allow for deeper insight into assesing the trade-off behavior, as for example, the effect of algorithm parameters.

4.2. Hypotheses

Any book or tutorial on experimentation probably agrees to the first necessary step in experimentation: Finding and formulating suitable questions (Montgomery, 2013; Moret, 2002). Following algorithm engineering principles, these questions are best stated

[3] Considering a line between each vertices of each dyad, the measure checks how many other vertices change their position such that they are on the other side of the line in the second layout.

[4] The authors do not mention any alignment of the two static layouts, which might be a reason for their findings of very large positional differences for this layout type.

as falsifiable hypotheses (Sanders, 2009). Not only is this closer to the gold standard that is the scientific method; the clear formulation of hypotheses directly implies factors and response variables to chose, and criteria for data collection or generation. Moreover, an explicit formulation of hypotheses prior to performing the experiment clarifies main goals and helps to resist the temptation to pick appropriate results a-posteriori as needed, and thus, makes experimental analysis more transparent and objective.

Regarding our evaluation of dynamic graph drawing by stress minimization, explicitly addressing stability by use of the presented methods instead of simply initializing with the preceding layout implies that a better compromise between quality and stability is expected. We break down assessment of this claim into constituent components to structure the discussion of detailed quantitative results in Section 4.4.

Our first hypothesis to test is thus that the methods actually display the assumed effects at all.

H 1 *Aggregation, anchoring, and linking increase dynamic stability, but reduce individual quality.*

Likewise, the explicit trade-off between quality and stability should be controllable via control parameter α.

H 2 *In anchoring and linking, higher values of α result in more stability and less quality.*

Being an iterative method, stress minimization is known to be susceptible to poor local minima and thus to depend on good initialization (Brandes and Pich, 2009). As a consequence, the same caveat should be in place where the outcome is not governed by the attempt to maintain stability.

H 3 *For decreasing values of control parameter α, anchoring and linking are increasingly sensitive to initialization.*

And finally, the principal adaptation to the offline scenario is by either anchoring to a reference position determined from the entire sequence of graphs, or by linking with future instance. These should pay off in cases where there is a persistent global structure.

H 4 *For dynamic graphs with persistent structure, anchoring to aggregate layouts and linking outperform online approaches.*

The experiments conducted in the following are designed to provide evidence for assessing these rather qualitative associations in detail.

4.3. Factors, response and experimental design

After stating suitable hypotheses, the next step is to fix the choice of factors, test instances, and response variables with respect to a randomized complete block design.

To be able to obtain detailed and generalizable insight into the behavior of our dynamic graph drawing approaches, it is particularly important to use realistic input graph sequences. In this section, we provide reasonable approaches to produce test instances, but we also address the issue of quantifying the output by relating measures to a lower bound derived from static stress minimization. Both, generation of meaningful test instances and obtaining strong lower bounds, are crucial to assess the behavior of heuristic approaches (Moret, 2002).

4.3.1. Data generation

The first three hypotheses stated in the previous section are very general with respect to data. As mentioned before, our focal application area are longitudinal social networks. Instead of using a (necessarily small) collection of benchmark networks, we generate random graphs that are believed to be realistic for the application scenario, because they are obtained from the two most prevalent models in this domain. The mechanism presented shortly can produce a large variety of structured sequences. To test H 4, we also need to employ a sequence data generator that produces non-structured sequences.

Although one could, in principal, control for several parameters for each data generator, many effects of the resulting sequences are likely not controllable. Thus, eventually, experiments will be blocked by those different sequence generators.

ERGM and SOAM The two models just mentioned are exponential-family random graph models (Robins et al., 2007; Hunter et al., 2008, *ERGM*), that model the characteristics of single networks, and stochastic actor-oriented models (Snijders, 2001; Snijders et al., 2010, *SAOM*), that model the evolution between two networks.[5] We will use ERGMs to create the initial graph of each sequence and SOAMs to obtain the actual sequence of graphs. Roughly speaking, both methods are based on network-specific characteristics, called *effects* or *statistics* – such as density of the network, reciprocity of edges for directed networks, or number of triangles – and associated model parameters determining whether an effect increases or decreases the probability of a network (ERGM), or of particular network changes (SAOM).

[5]Both methods are available as packages for the open source statistical system R (packages *ergm* and *RSiena*).

For ERGMs, the probability of a graph G from a certain class of graphs $\mathcal{G}$ belonging to a model specified by k effects $\gamma_i(G) : \mathcal{G} \to \mathbb{R}, 1 \leq i \leq k$, and parameters θ_i, is given by

$$P_\theta(G) = \frac{1}{\kappa(\theta)} \exp\left(\sum_{i=1}^{k} \theta_i \gamma_i(G)\right) ,$$

where $\kappa(\theta)$ is a normalizing factor. Given a graph and a set of effects, parameters can be estimated. When both are given, i.e., the model is completely specified, graphs can be generated via Markov-chain simulation.

Similar effects are used in SAOMs for modeling the evolution between two networks, which is regarded as a stochastic continuous-time Markov chain of elementary changes, called *micro-steps*, that are performed consecutively by randomly chosen vertices. At each step the probability of creation or deletion of an edge at a vertex v depends on its *objective function*

$$f_v(\theta, G) = \sum_{i=1}^{k} \theta_i \gamma_{i,v}(G) .$$

There are two main methods used for parameter estimation and simulation. The *method of moments* that estimates θ such that the expected value of statistics are equal to the actually observed statistics, and the *maximum likelihood* estimation that estimates θ maximizing the probability of actually observing the input data.[6]

Structured sequence generation A sequence of T graphs is created in the following way:

1. Two actual observations of a longitudinal network serve as the basis for the creation process. Let the corresponding graphs be G_1 and G_2. From the first network, we estimate an ERGM using the basic effects density, reciprocity of edges, outdegree, and edgewise shared partners, a measure related to transitivity.[7] Using this ERGM, an artificial first observation G_1^{sim} is simulated and made connected, by connecting a single random vertex of each component to a random vertex in the maximal component.

2. Next, a SAOM is estimated using the real observations G_1 and G_2, including effects for the the number of changes (parameterized by the so called *rate parameter*), outdegree, reciprocity, and transitivity.[8] The thus estimated SAOM is used for the following simulations.

[6] The maximum likelihood estimation in RSiena allows for retrieving the chain of micro-steps.
[7] These statistics are named `edges`, `mutual`, `gwodegree` and `gwesp` in the ergm package.
[8] Rate parameter, outdegree, and reciprocity are included in the RSiena model estimation by default. For transitivity, we use the effect `transitive triplets`.

3. The artificial second observation G_T^{sim} is obtained by running a simulation with the unconditional method of moments, using G_1^{sim} and G_2 as input. After ensuring connectedness of G_T^{sim} in the same way as for G_1^{sim}, a simulation with the maximum likelihood method using G_1^{sim} and G_T^{sim} as input is performed to obtain a reliable chain of micro-steps leading from G_1^{sim} to G_T^{sim}. This chain is partitioned into $T-1$ parts. Applying the corresponding changes to the initial observation G_1^{sim} yields a sequence of T networks.

As real input data, we use two data sets that are well studied in the social sciences.[9] The *s50* data set (Michell and Amos, 1997) comprises a sequence of three friendship networks of 50 female teenage pupils recorded over a three-year period. We use the first and third observation as input for network sequence generation. The second real data set used is the *van de Bunt* data set (Van De Bunt, Van Duijn, and Snijders, 1999), again, an evolving friendship network among 32 university freshmen comprising seven observations. Here, friendship is encoded ordinally. We obtain input for network sequence generation by only introducing edges with rating *best friendship* and *friendship* from the second and the seventh observation. The first observation was disregarded, since it only contained very few edges. Note also that we removed vertex 18, since it is isolated at all time points.

Figure 4.3 displays the R-code used to generate sequences from the van de Bunt dataset.[10] Methods `makeConnected` and `buildSequence` are not listed explicitly, but are straightforward to implement.

Unstructured Sequences In addition to the network generation process described above, we employ generation of more unstructured artificial data by means of the $G(n,p)$ random graph model (Gilbert, 1959). An initial observation is created with $n = 50$ and $p = \log(n)/n$, which produces connected graphs with high probability. Subsequently, $k/2$ edges are formed uniformly at random and, likewise, $k/2$ edges deleted, where we do not allow deletion of edges just formed. Afterwards, the resulting graph is made connected. This step is repeated until the desired number of graphs in the sequence is obtained. In our experiments, we use $k = 2\sqrt{n}$ and $k = n$.

4.3.2. Measurements

All our hypotheses require that we define suitable response variables describing the quality of individual layouts of a sequence, and the dynamic stability between two successive layouts of a sequence.

[9] Both data sets are publicly available on the Siena web page `http://www.stats.ox.ac.uk/~snijders/siena/`.

[10] The code was used with R version 2.12.2, RSiena version 1.0.12 and ergm version 2.3.

```
library(network); library(ergm); library(RSiena) # load libraries
### read data ###
G1 <- as.matrix(read.table("[path to first real observation]"))
G2 <- as.matrix(read.table("[path to last real observation]"))
net1 <- as.network(G1)

### ERGM model for first observation ###
model <- ergm(net1~edges+mutual+gwesp(0.1,fixed=T)+gwodegree(0.5,fixed=T))
# simulate artificial first observation (->GSim1)
netSim1 <- simulate.ergm(model, 1);
GSim1 <- as.matrix.network(netSim1,"adjacency")
# "make connected" connects a random vertex of each component to
# a random vertex of the largest component
GSim1 <- makeConnected(GSim1)

### Siena model estimation (G1->G2) ###
# set up model
mynet <- sienaNet(array(c(G1,G2), dim=c(n,n,2))) # n denotes number of vertices
mydata <- sienaDataCreate(mynet)
myeff <- getEffects(mydata)
# include transitivity parameter
myeff[myeff$effectName=='transitive triplets' & myeff$type=='eval', 'include'] <- TRUE
mymodel <- sienaModelCreate(findiff=FALSE,n3=1000,cond=F,projname='MoMEst')
# perform estimation
starvalue <- siena07(mymodel,data=mydata,effects=myeff,nbrNodes=2, useCluster=T, initC=
    T,clusterString=rep("localhost", 2),batch=T)
# starvalue now contains the estimated parameters

### Simulation of artificial second observation (->GSimT) ###
mynetSimT <- sienaNet(array(c(GSim1,G2), dim=c(n,n,2)))
mydataSimT <- sienaDataCreate(mynetSimT)
myeffSimT <- getEffects(mydataSimT)
myeffSimT[myeffSimT$effectName=='transitive triplets' &
  myeffSimT$type=='eval', 'include'] <- TRUE
# use previously estimated model
myeffSimT$initialValue[myeffSimT$include] <- starvalue$theta
mymodelSimT <- sienaModelCreate(projname = "MoMSim", useStdInits = F, n3 = 10, nsub =
    0,cond=F)
# perform simulation
simT <- siena07(mymodelSimT,data=mydataSimT,effects=myeffSimT, returnDeps=T,batch=T)
# retrieve first of the simulated networks (edge list format)
elSimT <- simT$sims[[1]][[1]][[1]][[1]]
# convert to adjacency matrix
GSimT <- as.matrix.network(as.network(elSimT),"adjacency")
GSimT <- makeConnected(GSimT)

### Simulate chain from artificial first to second observation ###
mynetSimC <- sienaNet(array(c(GSim1,GSimT), dim=c(n,n,2)))
mydataSimC <- sienaDataCreate(mynetSimC)
myeffSimC <- getEffects(mydataSimC)
myeffSimC[myeffSimC$effectName=='transitive triplets' & myeffSimC$type=='eval', '
    include'] <- TRUE
myeffSimC$initialValue[myeffSimC$include] <- starvalue$theta
mymodelSimC <- sienaModelCreate(projname = "MaxLikeSim", useStdInits = F, n3 = 10, nsub
     = 0, maxlike=T)
# perform simulation
simC <- siena07(mymodelSimC,data=mydataSimC,effects=myeffSimC, returnDeps=T,batch=T)
# create chain list
chain <- cbind(simC$f$chain[[1]]$Ego+1, simC$f$chain[[1]]$Alter+1)
# "build sequence" creates a sequence of t networks from the given chain
mysequence <- buildSeqence(chain, t, GSim1)
# mysequence[[i]] contains the adjacency matrix of graph i of the sequence
```

Figure 4.3.: R code for generating a sequence of networks from two observations of a real data set.

To assess the quality and stability of layouts of a dynamic graph, we use the measures that constitute our approaches, that are, stress and sum of squared positional difference. Although it may be doubted whether these measures really capture either quality or stability (Saffrey and Purchase, 2008), no other measures have been shown to better represent these concepts; it is therefore only reasonable to use the intrinsic measures of the approaches.

In the following, let P_M be the layout for a graph obtained by method M. Furthermore, let $\mathsf{stress}(P_M)$ be the stress of layout P_M as defined by Equation 2.2 and let

$$\varphi(P_M^{t-1}, P_M^t) = \sum_{i \in V} \left\| p_i^t - p_i^{t-1} \right\|^2$$

be the sum of squared positional difference between two subsequent layouts P_M^{t-1} and P_M^t, where these have been aligned by Procrustes analysis, that is, $\varphi(P_M^{t-1}, P_M^t)$ is minimal with respect to translation and rotation of P_M^{t-1} and P_M^t.

Raw measures A problem is that both measures are not directly comparable across graphs of different sizes or structure, and, more importantly, do not capture the ability of our methods to trade off quality for stability with respect to structural change between observations. Figure 4.4 illustrates the case for two observations generated from the van de Bunt data set and one observation generated from the s50 data set.

Regarding stress, we naturally notice the difference in scale for the different data sets, but also, values for the second observation from the van de Bunt data set are consistently higher than for the first observation. While we could apply normalization to stress, for example, by dividing by the sum of weights involved in the stress function (see Cox and Cox 2001 for this and other types of normalization), the measure still would be biased by the intrinsic stress of an observation. However, we rather want to asses the quality that is lost by introducing the various stability mechanisms compared to a layout with minimal stress not considering stability at all.

The case is even more severe for positional differences. Similar to stress, we notice a largely different scale between the two data sets due to different sizes, which could be normalized by the number of dyads of the sum. However, the major problem is the difference in scale within two transitions of a sequence, where certain structural changes must result in much larger movement than others. Consider, for example, the deletion of a single edge from a path connecting two vertices. If there are more similar-length paths connecting the same two vertices, the distance matrix does not change considerably, and thus, the layout will be very similar. On the other hand, if this path was the only connection of similar length between those vertices, distances change drastically, and therefore, must result in a severe change of the layout to faithfully represent distances.

Therefore, the ability to trade off quality for stability can not be assessed by the raw measure of positional difference, but again, has to be related to a worst-case scenario, that is, the transition between the two layouts with minimal stress.

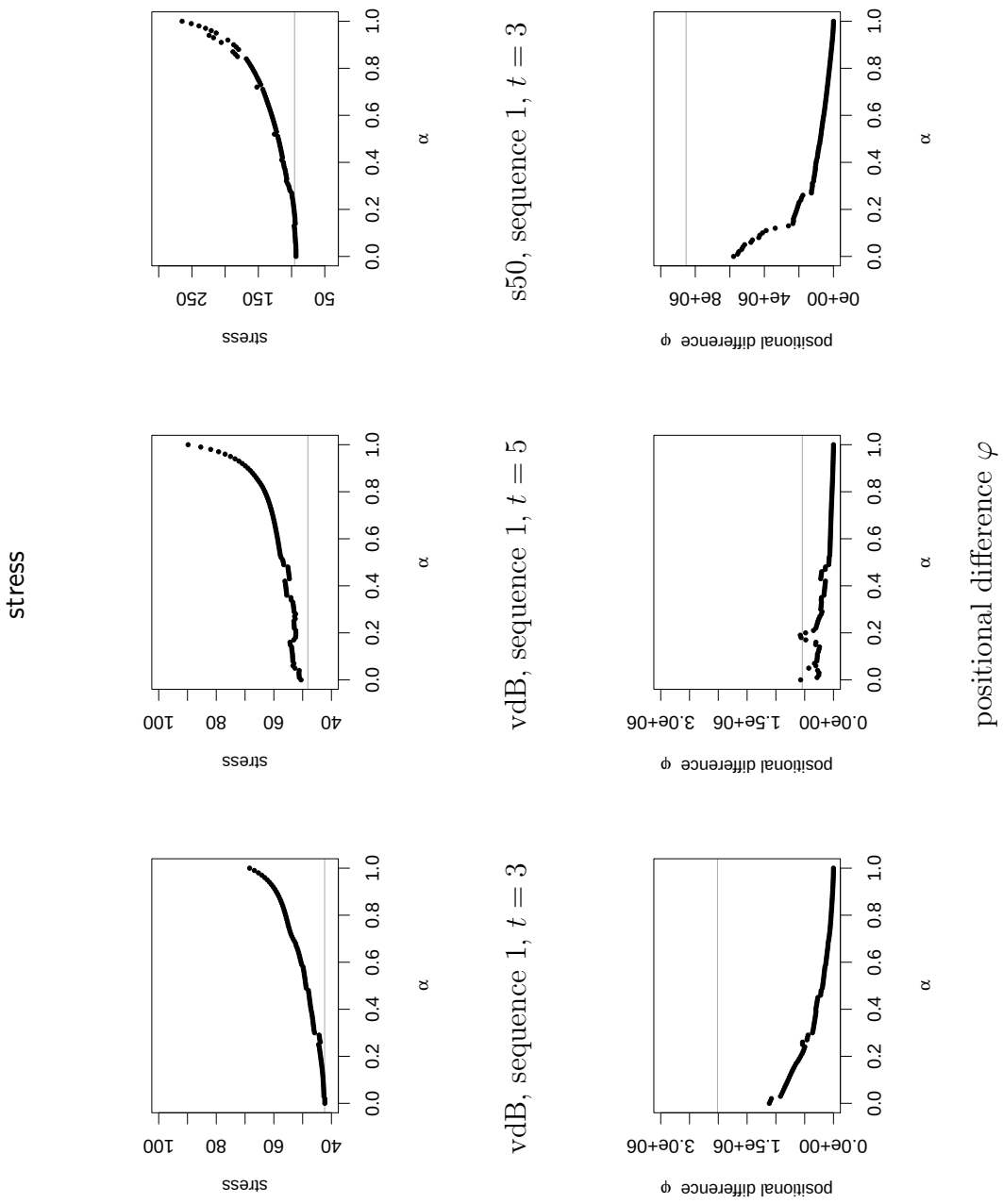


Figure 4.4.: Raw measurements of stress and positional difference φ subject to trade-off parameter α for two observations generated from the van de Bunt data set and one observation generated from the s50 data set (layout by stress^{APA}). Raw measures are not directly comparable for different graphs due to different intrinsic stress and positional difference imposed by different size, structure and structural change. Gray horizontal lines indicate values of the baseline solution.

Relating measures to static stress We have argued in Section 2.3, that it is hard to obtain a static layout with minimal stress, which we need as a baseline to compare raw measures to. However, the approach of minimizing stress after initializing with CMDS usually results in low-stress layouts, and therefore, is a natural choice here.

We thus relate both measures to the ones obtained from a common baseline method **B**, that is, we compute static layouts for each graph in the sequence as suggested in Section 2.3, and align the sequence by Procrustes analysis after layout calculation. The baseline values for our measurements are depicted as gray horizontal lines in Figure 4.4.

We measure quality δ_σ^M for method M as the fraction of $\mathsf{stress}(P_B)$ and $\mathsf{stress}(P_M)$, i.e.,

$$\delta_\sigma^M = \frac{\mathsf{stress}(P_B)}{\mathsf{stress}(P_M)} \ .$$

Since we assume that the baseline layout is of relatively high quality, and that quality will suffer, whenever mechanisms to increase stability are employed, the range of δ_σ^M should be $(0, 1]$, with δ_σ^M decreasing for higher values a.

Stability is measured as the relative decrease of positional difference w.r.t. the baseline layout, i.e.,

$$\delta_\varphi^M = 1 - \frac{\varphi(P_M^{t-1}, P_M^t)}{\varphi(P_B^{t-1}, P_B^t)}$$

The assumption is that the baseline layout exhibits a high positional difference, that will decrease whenever mechanisms to increase stability are employed. Note that all methods presented yield the same layout for all graphs in the sequence for $\alpha = 1$, therefore δ_φ^M must be 1 for all methods in this case. Thus the range is expected to be $[0, 1]$, with δ_φ^M increasing for higher values α.

Figure 4.5 shows the measures for quality and stability as defined above for the same observations of Figure 4.4. For the two observations of the van de Bunt data set, we observe that both measures are now less biased with respect to the intrinsic stress of an individual observation, and intrinsic need for movement of a transition, respectively, and thus are comparable.

4.3.3. Performing the experiment

For each network sequence generator, we created 50 network sequences comprising 10 graphs each. We measured δ_σ and δ_φ corresponding to trade-off parameter $\alpha \in \{0, 0.01, 0.02, \ldots, 1\}$. Thus, per generator, method, and value of α, we obtain 500 measurements of δ_σ (450 for **APP** and **ACP**, since for the first observation of each sequence in these cases $\delta_\sigma = 1$ for all α), and 450 measurements of δ_φ (not applicable to each first observation). Figure 4.6 shows five-point summaries, i.e., minimum, first quartile, median, third quartile, and maximum, of the measurements obtained for the 50 network sequences generated from the s50 data set when applying the **APP** and **APA** methods.

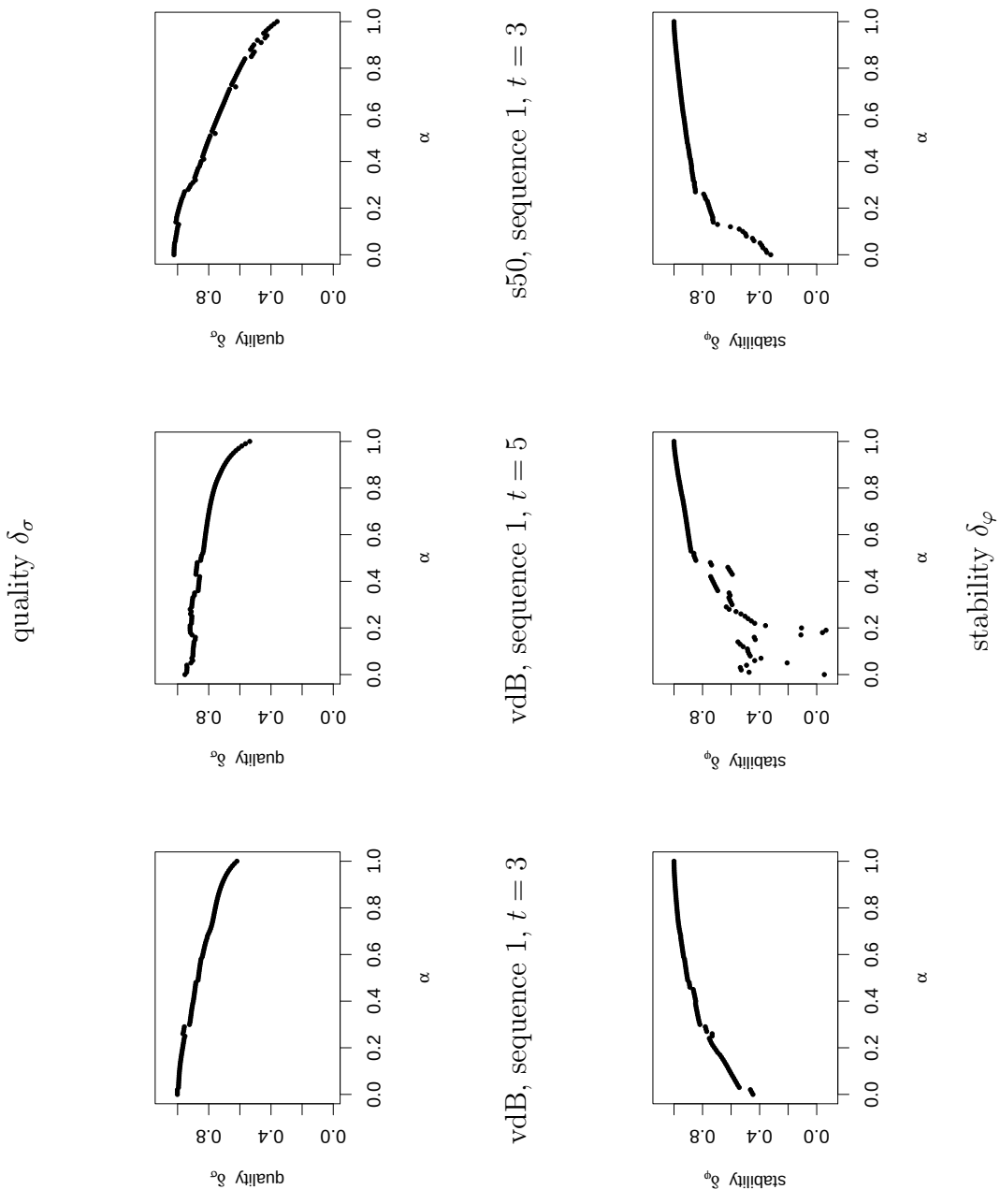
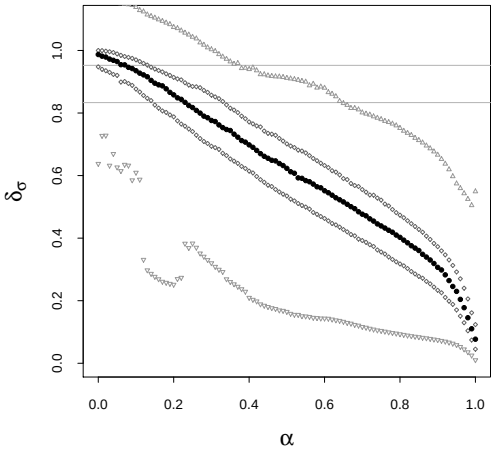
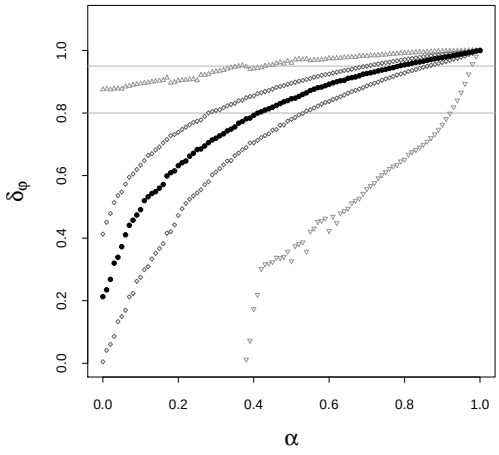


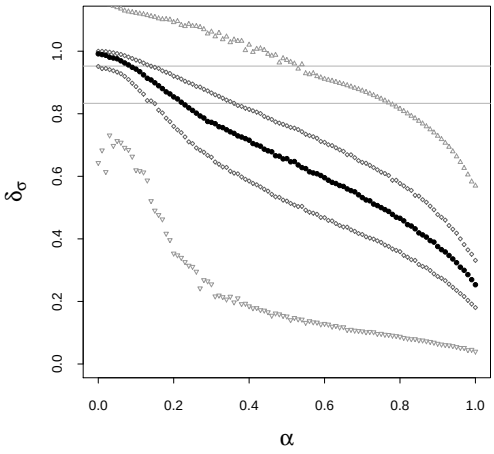
Figure 4.5.: Measurements relative to the baseline layout: quality δ_σ and stability δ_φ subject to trade-off parameter α for two observations generated from the van de Bunt data set and one observation generated from the s50 data set (layout by stress^{APA}). Relating raw measures to the baseline layout serves as normalization for graphs of different size, structure, and structural change.



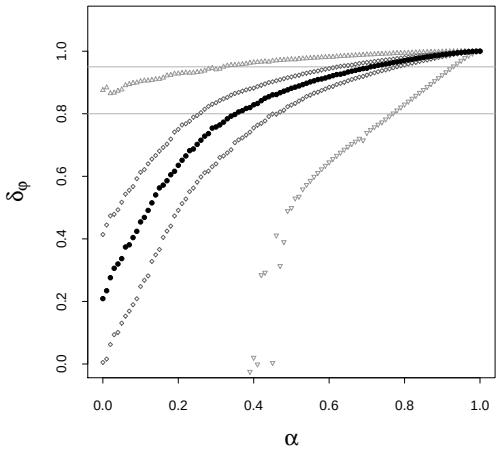
(a) s50 – **APP** – quality



(b) s50 – **APP** – stability



(c) s50 – **APA** – quality



(d) s50 – **APA** – stability

Figure 4.6.: Five-point summary for measurements of δ_σ and δ_φ subject to trade-off parameter α with methods **APP** and **APA** for 50 network sequences generated from the s50 data set. Gray horizontal lines indicate thresholds used in the main text.

Appendix A contains five-point summaries for all data sets and methods. The gray horizontal lines indicate thresholds used in our experiments, that are, 5% and 20% more stress w.r.t. the baseline for quality measurements (from top to bottom), and 80% and 95% reduction in positional difference w.r.t. to the baseline for stability measurements (from bottom to top).

Note that, for both δ_σ and δ_φ, there are outliers that contradict the intuitive assumptions regarding the range of the measures. We can only explain these by the heuristic nature of stress minimization. Still, most of the measured values are within a reasonable range around the median values, as can be observed by the inter-quartile range.

Thus, we will argue about the approaches by means of the median measurements, denoted by $\hat{\delta}_\sigma$ and $\hat{\delta}_\varphi$, respectively. Table 4.2 lists the measured median values of δ_σ and δ_φ at selected levels of $\alpha = (0.1, 0.2, 0.3, 0.7, 0.8, 0.9)$, for all network sequence generators and layout variants. If it is guaranteed that a difference between two values is not due to rounding, that is, if the range spans more than 0.02, the two highest value for each block and parameter α are highlighted.

In order to remove dependance on the trade-off parameter α, we also juxtapose median values for δ_φ at selected median values of δ_σ, and vice versa. Table 4.3 summarizes these measurements for all data sets and methods, and Figure 4.7 provides a graphical overview. Each left (right) endpoint in the upper row of Figure 4.7 shows the median value of δ_φ corresponding to the value α where the median value of δ_σ was last greater or equal to 1/1.2 (1/1.05), i.e., where stress did not increase by more than 20% (5%) compared to the baseline. Analogously, each left (right) endpoint in the lower row displays the median value of δ_σ, where reduction in positional difference was first greater or equal to 80% (95%).

For example, to obtain the median value of δ_φ at $\delta_\sigma = 1/1.2$ for the s50 data set and method **APP** (the left red circle in the upper-left plot from Figure 4.7), we check the value of α, at which δ_σ last is higher than the threshold (roughly 0.25 in Figure 4.6a), and report the corresponding median value of $\delta_\varphi = 0.67$ at this value of α (see Figure 4.6b).

4.4. Results

Having fixed a reasonable experimental design, and having performed the experiments, the last step of the experimental study is to analyze the results and draw conclusions regarding the original objectives.

H 1: *Aggregation, anchoring, and linking increase dynamic stability, but reduce individual quality.*

Figure 4.7 (upper row, right endpoints) shows that already a slight compromise in quality (5% additional stress compared to static baseline layouts) yields a large increase in positional stability (ranging from 24% to 82% reduction of total movement) for methods with explicit control for stability. If we allow a 20% increase of stress (left-hand side of each upper graph), all methods reduce movement by more than 50%.

The aggregate layout always yields maximum stability in terms of positional difference. Since approaches that anchor to the aggregate layout exactly result in the aggregate layout at parameter $\alpha = 1$, we can observe from Figure 4.6c and corresponding ones in Appendix A, that the quality of the aggregate layout is usually significantly lower compared to the baseline.

There are, however outliers to the general trend. We can observe that, at lower ranges of parameter α, there are instances that provide higher individual quality in terms of stress than the baseline layout (data points above value 1 on the y-axis in Figure 4.6a), and instances that exhibit lower dynamic stability in terms of positional difference than the baseline layout (data points below value 0 on the y-axis in Figure 4.6b), even though stability mechanisms are employed. An explanation is, that the baseline layout is only an approximation of an "optimal" layout, and that changes in the objective function via stability terms and different initialization sometimes yield a local optimum that is better for individual quality and/or worse for dynamic stability.

Still, the experiment provides evidence that the methods are largely having the desired effect of increasing dynamic stability at the cost of individual layout quality.

H 2: *In anchoring and linking, higher values of α result in more stability and less quality.*

Table 4.2 shows median values of the quality measure δ_σ and the stability measure δ_φ at selected levels of α. With increasing value of parameter α, we observe strictly monotonically decreasing median values of δ_σ, and strictly monotonically increasing median values of δ_φ, for any data set and method. Moreover, a closer look at Figure 4.6 (and Figures in Appendix A) reveals that median and inter-quartile bands of both measures generally develop in a monotonic way, as expected.

	$\hat{\delta}_\sigma$ at $\alpha =$	APP	ACP	APA	ACA	LAG	LCG	LAW	LCW
s50									
	0.1	0.94	0.97	0.94	0.96	0.96	0.98	0.95	0.97
	0.2	0.86	0.90	0.86	0.87	0.91	0.93	0.90	0.92
	0.3	0.78	0.80	0.77	0.79	0.86	0.89	0.85	0.87
	⋮								
	0.7	0.48	0.48	0.54	0.53	0.68	0.69	0.68	0.68
	0.8	0.40	0.40	0.47	0.47	0.62	0.63	0.62	0.63
	0.9	0.31	0.31	0.37	0.38	0.54	0.54	0.55	0.55
vdB									
	0.1	0.97	1.00	0.98	0.99	0.98	1.00	0.97	0.99
	0.2	0.94	0.98	0.95	0.97	0.97	0.99	0.96	0.98
	0.3	0.92	0.95	0.93	0.94	0.95	0.97	0.95	0.96
	⋮								
	0.7	0.75	0.75	0.81	0.82	0.89	0.89	0.89	0.89
	0.8	0.69	0.69	0.78	0.77	0.86	0.86	0.86	0.87
	0.9	0.62	0.62	0.72	0.72	0.83	0.82	0.83	0.83
$G(n,p), k = 2\sqrt{n}$									
	0.1	0.97	0.99	0.98	0.99	0.98	0.99	0.98	0.99
	0.2	0.94	0.96	0.94	0.96	0.96	0.98	0.96	0.97
	0.3	0.89	0.92	0.91	0.92	0.95	0.96	0.94	0.95
	⋮								
	0.7	0.71	0.71	0.77	0.77	0.86	0.86	0.86	0.86
	0.8	0.65	0.66	0.72	0.72	0.83	0.82	0.83	0.83
	0.9	0.57	0.58	0.66	0.66	0.78	0.77	0.79	0.78
$G(n,p), k = n$									
	0.1	0.95	0.98	0.96	0.98	0.96	0.99	0.95	0.98
	0.2	0.89	0.93	0.90	0.93	0.91	0.95	0.89	0.93
	0.3	0.82	0.84	0.83	0.85	0.86	0.90	0.85	0.88
	⋮								
	0.7	0.53	0.53	0.60	0.60	0.69	0.69	0.69	0.68
	0.8	0.47	0.47	0.54	0.54	0.64	0.64	0.64	0.64
	0.9	0.40	0.40	0.47	0.48	0.58	0.57	0.58	0.57

Table 4.2.: Median values for quality δ_σ at certain selected values of α for all measurements. Note that only measurements belonging to either the anchoring or linking approaches can be compared directly, due to systematic differences. If the range of values spans more than 0.02, the two highest values are colored.

	$\hat{\delta}_\varphi$ at $\alpha =$	APP	ACP	APA	ACA	LAG	LCG	LAW	LCW
s50									
	0.1	0.49	0.29	0.46	0.22	0.51	0.35	0.57	0.43
	0.2	0.63	0.54	0.64	0.52	0.69	0.59	0.73	0.64
	0.3	0.72	0.70	0.76	0.72	0.78	0.72	0.81	0.75
	⋮								
	0.7	0.93	0.93	0.95	0.95	0.95	0.94	0.95	0.94
	0.8	0.95	0.95	0.97	0.97	0.97	0.97	0.97	0.97
	0.9	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
vdB									
	0.1	0.58	0.21	0.61	0.16	0.61	0.26	0.66	0.33
	0.2	0.70	0.44	0.71	0.39	0.75	0.48	0.78	0.56
	0.3	0.77	0.65	0.80	0.65	0.82	0.66	0.84	0.71
	⋮								
	0.7	0.94	0.94	0.96	0.96	0.96	0.94	0.96	0.94
	0.8	0.96	0.96	0.98	0.98	0.98	0.97	0.97	0.97
	0.9	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
$G(n,p), k = 2\sqrt{n}$									
	0.1	0.52	0.30	0.55	0.24	0.58	0.36	0.63	0.44
	0.2	0.67	0.57	0.69	0.54	0.73	0.60	0.76	0.65
	0.3	0.76	0.73	0.79	0.75	0.81	0.74	0.83	0.77
	⋮								
	0.7	0.95	0.95	0.96	0.96	0.96	0.95	0.96	0.95
	0.8	0.97	0.97	0.98	0.98	0.98	0.97	0.98	0.97
	0.9	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
$G(n,p), k = n$									
	0.1	0.40	0.20	0.34	0.13	0.40	0.25	0.48	0.34
	0.2	0.57	0.43	0.51	0.32	0.61	0.49	0.68	0.58
	0.3	0.69	0.63	0.65	0.58	0.74	0.67	0.79	0.73
	⋮								
	0.7	0.95	0.95	0.94	0.94	0.95	0.94	0.96	0.95
	0.8	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	0.9	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 4.2.: Median values for stability δ_φ at certain selected values of α for all measurements. Note that only measurements belonging to either the anchoring or linking approaches can be compared directly, due to systematic differences. If the range of values spans more than 0.02, the two highest values are colored.

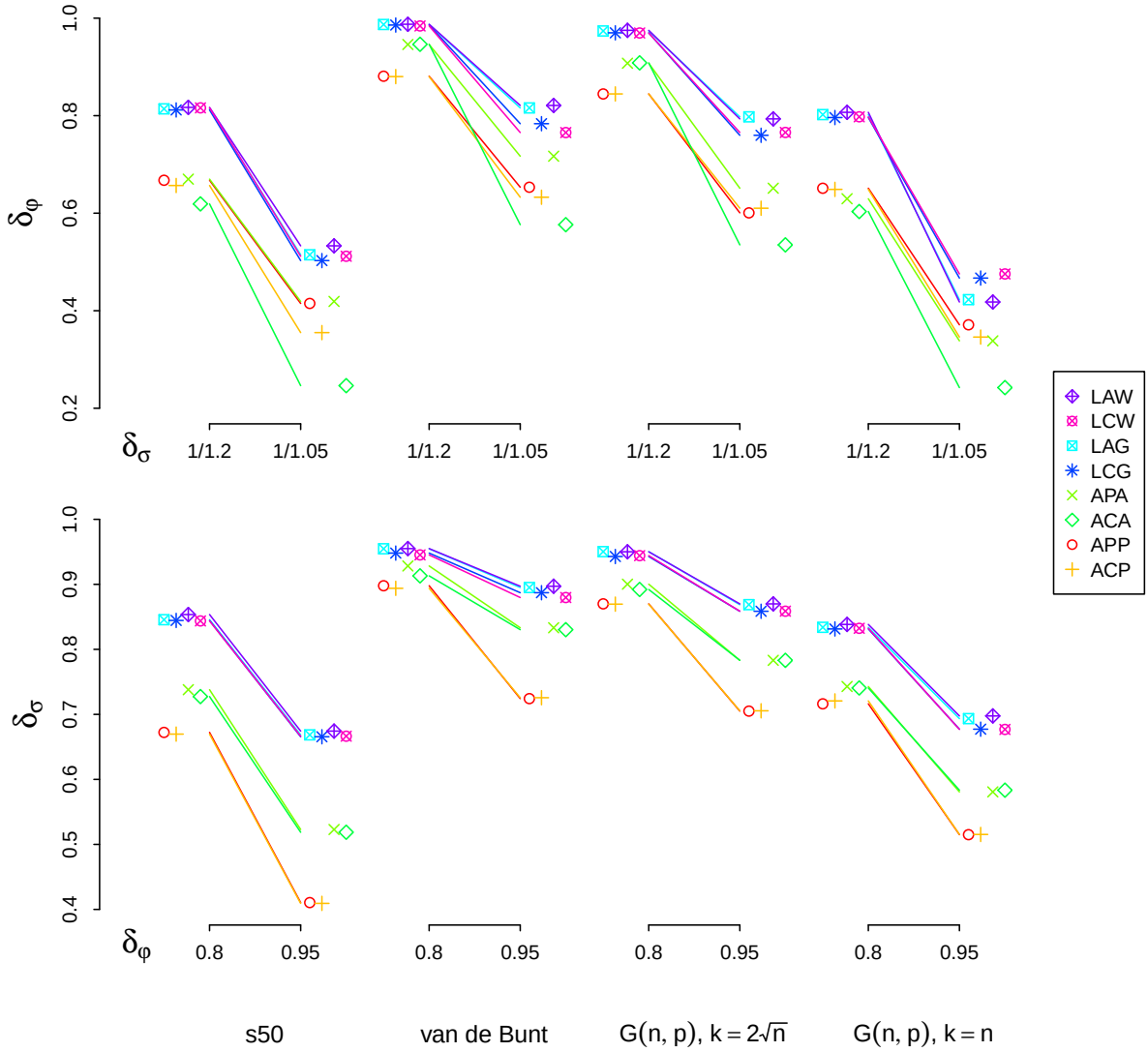


Figure 4.7.: Graphical overview of median values for δ_φ at selected median values of δ_σ (upper part) and vice versa (lower part) for all measurements. The four columns correspond to data sets s50, van de Bunt, $G(n,p), k = 2\sqrt{n}$, and $G(n,p), k = n$ (from left to right). For exact values, see Table 4.3.

		APP	ACP	APA	ACA	LAG	LCG	LAW	LCW
	$\hat{\delta}_\varphi$ at $\hat{\delta}_\sigma =$								
s50									
	1/1.2	0.67	0.66	0.67	0.62	0.81	0.81	0.82	0.82
	1/1.05	0.41	0.36	0.42	0.25	0.51	0.50	0.53	0.51
vdB									
	1/1.2	0.88	0.88	0.95	0.95	0.99	0.99	0.99	0.98
	1/1.05	0.65	0.63	0.72	0.58	0.82	0.78	0.82	0.77
$G(n,p), k = 2\sqrt{n}$									
	1/1.2	0.84	0.84	0.91	0.91	0.97	0.97	0.98	0.97
	1/1.05	0.60	0.61	0.65	0.54	0.80	0.76	0.79	0.77
$G(n,p), k = n$									
	1/1.2	0.65	0.65	0.63	0.60	0.80	0.80	0.81	0.80
	1/1.05	0.37	0.35	0.34	0.24	0.42	0.47	0.42	0.48
	$\hat{\delta}_\sigma$ at $\hat{\delta}_\varphi =$								
s50									
	0.8	0.67	0.67	0.74	0.73	0.85	0.84	0.85	0.84
	0.95	0.41	0.41	0.52	0.52	0.67	0.67	0.67	0.67
vdB									
	0.8	0.90	0.89	0.93	0.91	0.95	0.95	0.95	0.95
	0.95	0.72	0.73	0.83	0.83	0.90	0.89	0.90	0.88
$G(n,p), k = 2\sqrt{n}$									
	0.8	0.87	0.87	0.90	0.89	0.95	0.94	0.95	0.94
	0.95	0.71	0.71	0.78	0.78	0.87	0.86	0.87	0.86
$G(n,p), k = n$									
	0.8	0.72	0.72	0.74	0.74	0.83	0.83	0.84	0.83
	0.95	0.51	0.52	0.58	0.58	0.69	0.68	0.70	0.68

Table 4.3.: Median values for δ_φ at certain median values of δ_σ and vice versa for all measurements. For a graphical overview, see Figure 4.7.

However, looking back at measures of quality and stability for individual network instances of Figure 4.5, we observe that the corresponding curves are not necessarily smooth, and thus, locally contradict the hypothesis. It stands to reason that this effect is, again, due to the heuristic nature of stress minimization, and caused by a change of the local optimum due to change of the objective function, when slightly increasing parameter α.

Nevertheless, the experiment supports that generally parameter α indeed controls the trade-off between quality and stability in the expected way, with the slight caveat of local inconsistencies.

As a side note, we observe that reduction in positional difference usually increases very rapidly at lower ranges of α, similar to Figure 4.6. This observation corroborates earlier findings from the user study by Purchase and Samra (2008) that low stability mechanisms appear to be the most effective ones, see Section 4.1. However, it also raises the question, whether this is an intrinsic effect of the layout algorithm, rather than one of human perception.

H 3: *For decreasing values of control parameter α, anchoring and linking are increasingly sensitive to initialization.*

At lower ranges of parameter α ($\alpha \leq 0.3$), Table 4.2 reveals consistent differences in quality δ_σ in favor of initialization with classical scaling. However, the range of values is at most 0.05, that is, the difference is not considerable. Contrarily, there are significant differences in stability in favor of initialization with the previous layout for anchoring approaches, or the aggregate layout for linking approaches, for these low ranges of α, as expected.

At higher range ($\alpha \geq 0.7$), we observe very similar results for quality δ_σ for each two anchoring approaches belonging together with respect to the choice of reference. For the linking approaches, all variants behave very similar with respect to quality measures, which is also clearly visible from the lower graphs in Figure 4.7. Moreover, both anchoring and linking variants all yield roughly the same value regarding stability at these higher ranges of parameter α.

Thus, the experiment supports that initialization plays a significant role at lower ranges of control parameter α, which vanishes when the parameter is increasing.

H 4: *For dynamic graphs with persistent structure, anchoring to aggregate layouts and linking outperform online approaches.*

Regarding H 4, we first posit that data generated from ERGMs and SAOM learned from real data contains more persistent structures, since change in these networks is actually based on structural processes unlike the random changes in the $G(n, p)$ generator. To support the hypotheses, we would expect to see a change in the ranking of effectiveness between online approaches (anchoring approaches with the previous layout as reference – **A?A**) and offline approaches (anchoring approaches with the aggregate layout as reference – **A?P**, and all linking approaches) between the two data sets generated by realistic models, and the two data sets based on the $G(n, p)$ generator.

Rather surprisingly, though, at low stability levels (stress increase of 5% and 20%), we cannot observe any clear general trend in Fig. 4.7 that **A?A** approaches perform better than **A?P** approaches between the two types of generators. The only consistent observation is that method **ACA** performs worse, which indicates that initialization with classical scaling is in conflict with anchoring to the aggregate.

However, the more stability is sought, the more the difference in increase of stress between the **A?A** and the **A?P** approaches, in favor of the former, and regardless of the data set. This indicates that, with higher stability requirements, offline anchoring is more effective regardless of how structured the input is.

The linking approaches perform better than the anchoring approaches throughout. To our surprise, the choice between the two functions ζ_G and ζ_W does not seem to influence the results.

Thus, our evaluation of H 4 is inconclusive, since no statement can be made for low stability, and methods incorporating offline information (with the exception of **ACA**) apparently perform better regardless of structure for moderate, and especially, high stability requirements.

The experiment suggests that, to evaluate this hypothesis, a deeper look into the data generation mechanisms is needed. For example, ERGM and SOAM models that employ more specific structural effects and include behavioral effects might lead to sequences that allow stronger assumptions regarding change effects.

5. Conclusion

This thesis treated a cornerstone for explorative visualization of dynamic networks, namely, computing a sequence of straight-line graph layouts, corresponding to a given input sequence of general graphs. The key difficulty of this problem is to explicitly control the trade-off between quality of individual layouts and coherence of layouts in the sequence. We here adopted the most prominent paradigm to address coherence, which is catering for stability between observations by minimizing positional difference between vertex instances of the same vertex.

The main contributions of this thesis are summarized next.

- We categorized previous work following the paradigm of positional stability into three principal approaches and, correspondingly, presented consistent adaptations of stress minimization, the state-of-the-art method to produce static straight-line layouts for general graphs. This three approaches are aggregation, where vertices are at the same position throughout the sequence of layouts; anchoring, where vertices are attracted by a reference position; and linking, where vertices are attracted by positions of their copies in neighboring observations.

  Based on different initialization and reference type, we proposed variations for online and offline dynamic graph drawing, and qualitatively discussed characteristic differences. In summary, offline methods serve best to visually explore the evolution of a dynamic network. Aggregation facilitates getting an overview of global structure, whereas anchoring (with a aggregation as reference) and linking provide explicit control over the balance of readability and stability, and therefore allow to put emphasis on the specifics of individual networks in the sequence, while avoiding unnecessary flips of substructures and reducing excessive vertex movement.

- The consistent embedding in a single framework allowed us to quantitatively evaluate variant approaches in a novel way. We employed a hypothesis-based experimental design following algorithm engineering principles. To be able to test our hypotheses, we related measures for quality (in terms of stress) and stability (in terms of vertex movement) to a baseline determined from Procrustes-aligned static layouts, to normalize over graphs of different sizes and structure. A further novel aspect is our use of more sophisticated graph generators that eliminate reliance on small benchmark data sets and still produce application-typical data. Concretely, ERGMs were used for boundary observations and SAOMs for the evolution.

Our results supported the hypotheses that the proposed methods indeed trade off quality for increased stability and that this effect generally is controllable via the trade-off parameter. However, our experiments also revealed that this desired behavior does not always pertain locally. We also could confirm sensitivity to initialization for low stability situations. Interestingly, we could neither support or falsify our assumption that offline variants perform better than online variants on dynamic graphs with a persistent global structure, but found that linking is a generally preferable approach.

Our admittedly narrow focus on minimalistic formulations of dynamic stress variants following the paradigm of positional stability, and experimental evaluation by the two particular intrinsic measures for quality and stability, allowed us to compare these variants and to explain principal behavior. Important avenues to further our algorithmic experiments include evaluation of more detailed hypotheses, for example, regarding outliers or specific structures of the input sequence. Moreover, with ongoing expert analysis of real networks through the presented random graph models, and improvement of these models, our data generation procedures can surely be refined, for example, by including behavioral effects.

This, of course, is not the end of the story. We hold that such algorithmic experimentation on dynamic stress minimization, and graph drawing algorithms in general, fills a previously existing gap in characterizing those algorithms, and helps in designing and controlling further human interaction experiments. There are several directions for future work that certainly would benefit from an integral cycle of theoretical analysis, qualitative discussion, quantitative algorithmic experimentation and user studies:

- Is positional stability the best stability model? As mentioned, there are several more candidate difference metrics available. Most of these could be integrated in the stress minimization framework, and thus, be tested for differential characteristics.
- Can the trade-off between quality and stability be improved by engineering stability weights in our formulations in a sophisticated way? Such considerations might include analyzing the input sequence for specific graph structures and change patterns.
- Related to the two previous questions, how can a dynamic network be dealt with that does not exhibit a consistent overall global structure, for example, because observations were taken over a long time-span? Are there ways to preprocess such networks, use specific stability models or integrate other measures to the layout algorithm?
- The methods presented here are probably best suited in the stage where a researcher explores data. However, an adaption of stress minimization to substance-based visualizations, like centrality drawings, where a certain dimension of the layout space is used to represent and analyze information different from graph

structure, seems straight-forward. What are the implications of this reduction of the degree of freedom on the trade-off between quality and stability?

- We here treated the logical update, but there is much room for improvement for the physical update beyond the decision of using small multiples or animation to represent the dynamic network. Both types could be refined by sophisticated interaction mechanisms, or potentially, by displaying cues on changes that happened or will happen in the network through graphical annotations.

A. Appendix

Measurement five-point summaries

We here include complete five-point summaries of measurements for all data sets and methods evaluated in Chapter 4. Lines correspond to minimum, first quartile, median, third quartile, and maximum observation per value of trade-off parameter α.

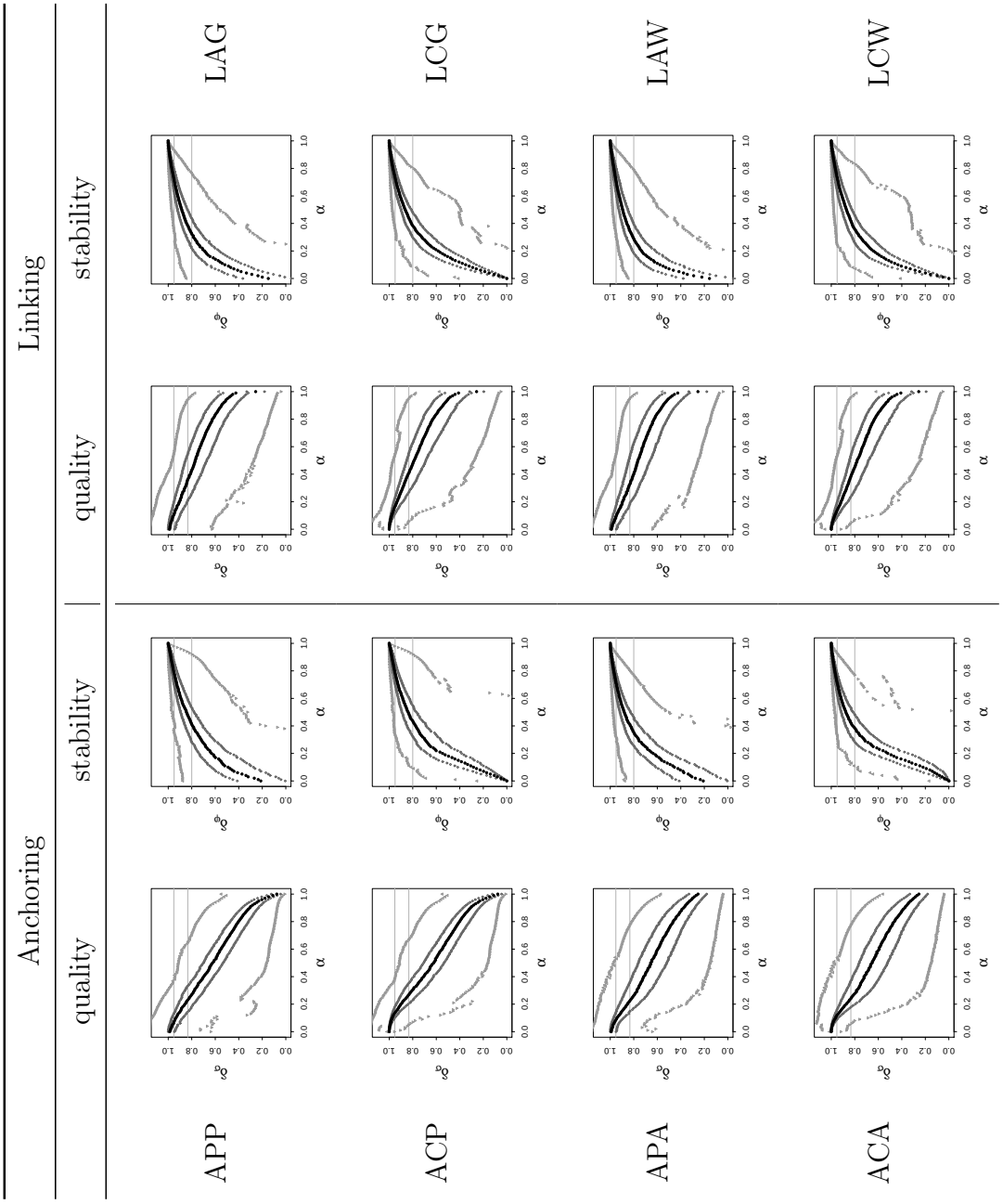


Table A.1.: Five-point summaries of measurements of δ_σ and δ_φ subject to trade-off parameter α for all methods evaluated on the **s50** data set. Grey horizontal lines indicate thresholds used in the evaluation.

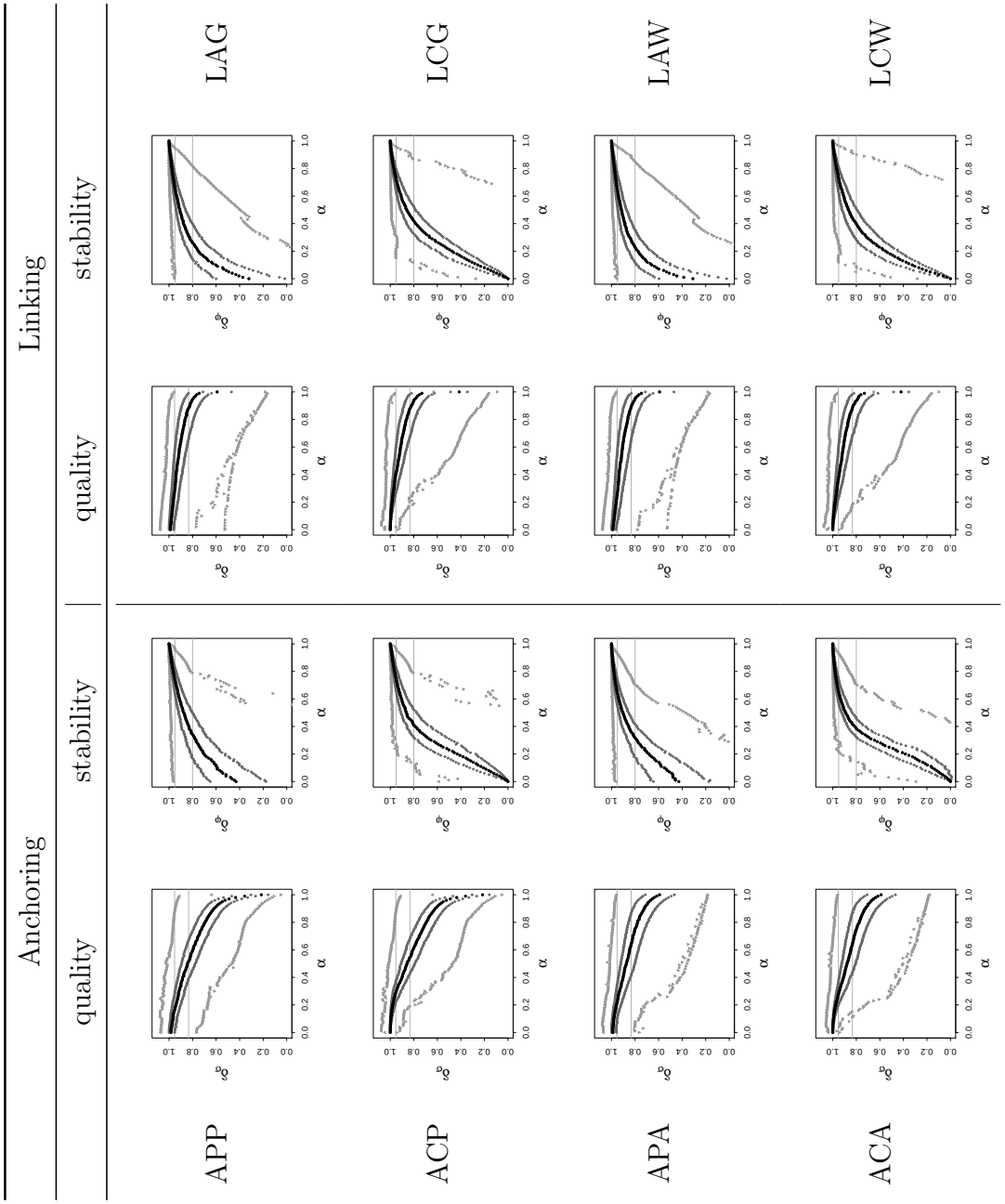


Table A.2.: Five-point summaries of measurements of δ_σ and δ_φ subject to trade-off parameter α for all methods evaluated on the **van de Bunt** data set. Grey horizontal lines indicate thresholds used in the evaluation.

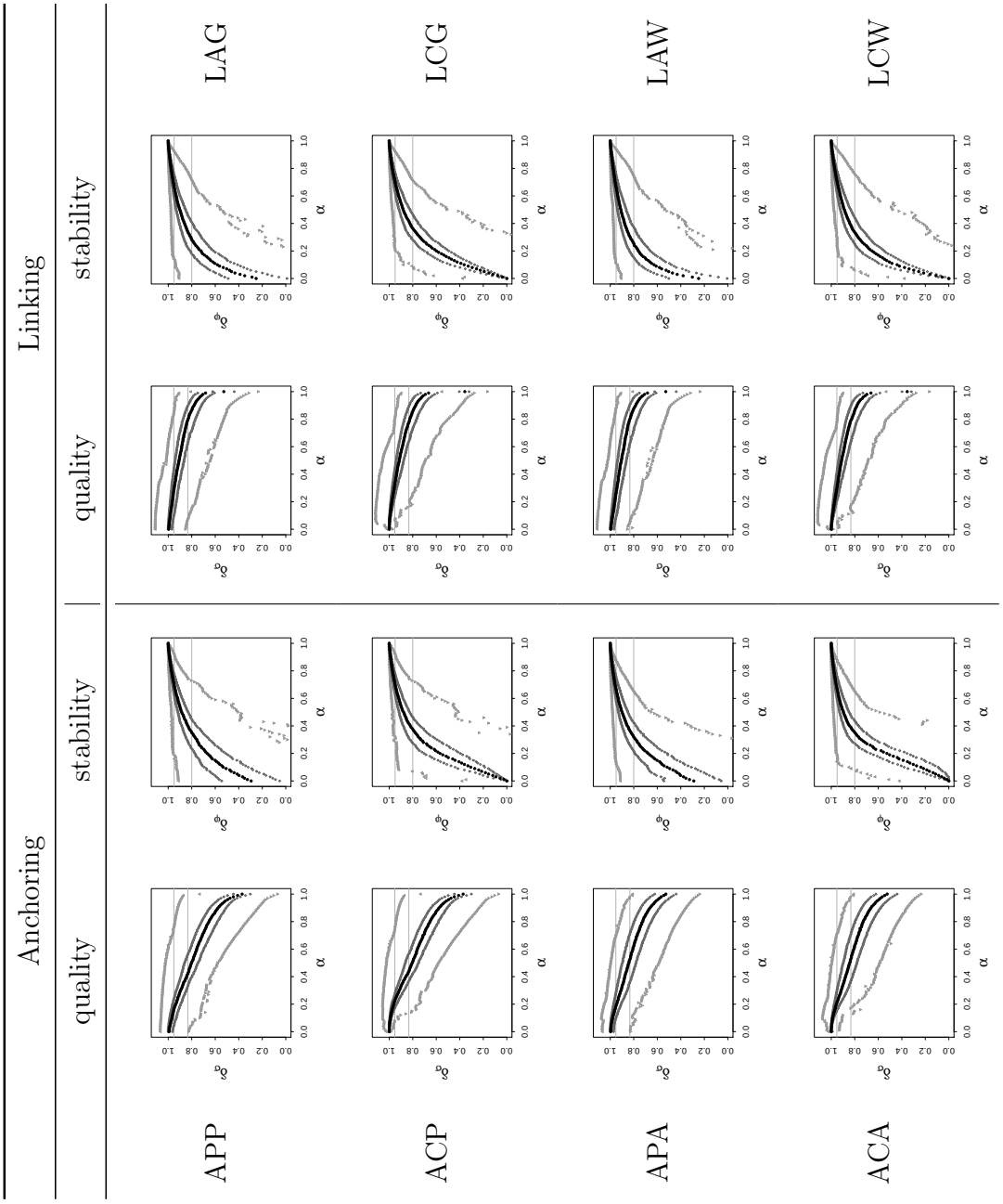


Table A.3.: Five-point summaries of measurements of δ_σ and δ_φ subject to trade-off parameter α for all methods evaluated on the $\mathbf{G}(\mathbf{n}, \mathbf{p}), \mathbf{k} = \mathbf{2}\sqrt{\mathbf{n}}$, data set. Grey horizontal lines indicate thresholds used in the evaluation.

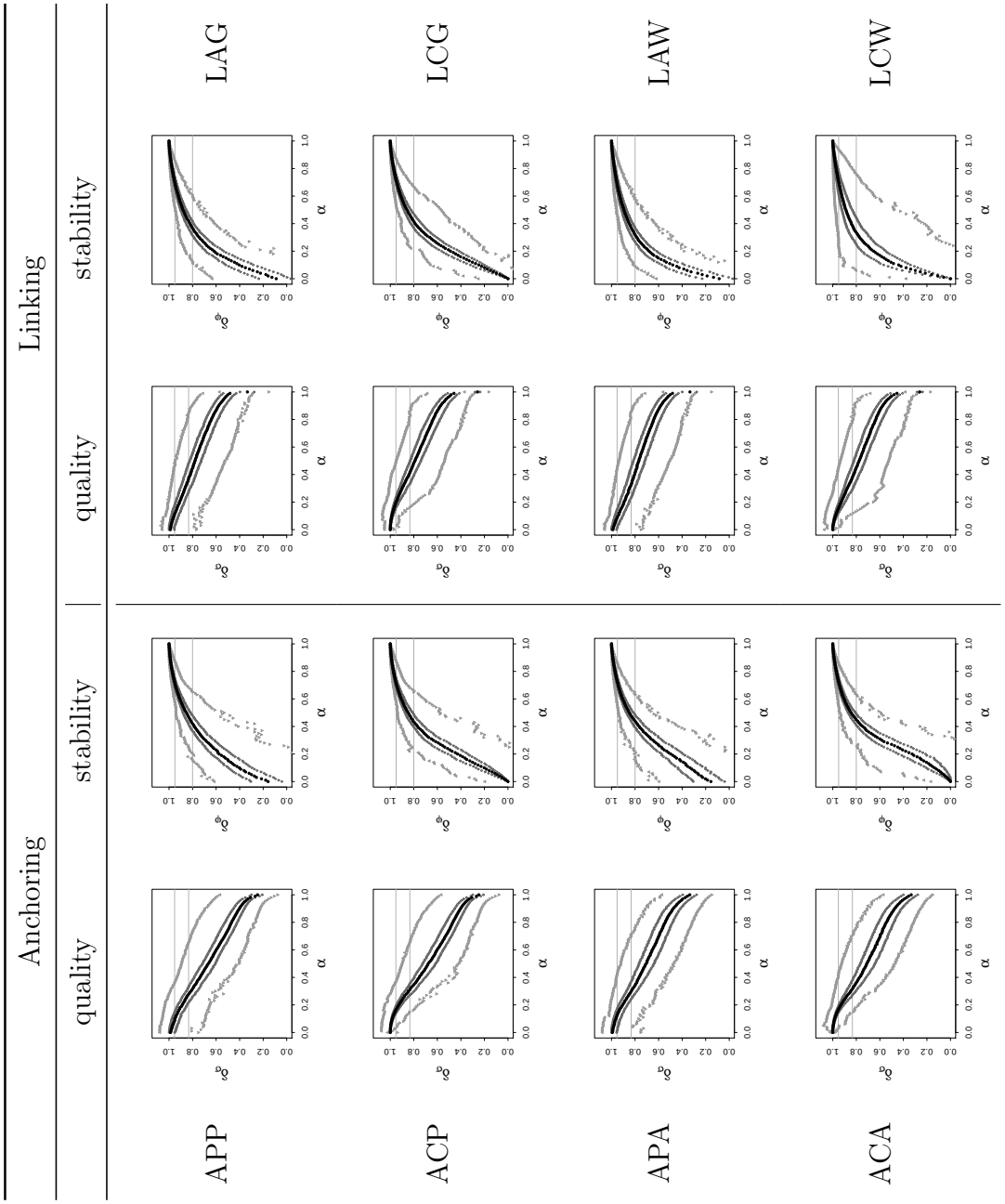


Table A.4.: Five-point summaries of measurements of δ_σ and δ_φ subject to trade-off parameter α for all methods evaluated on the $\mathbf{G}(\mathbf{n}, \mathbf{p}), \mathbf{k} = \mathbf{n}$, data set. Grey horizontal lines indicate thresholds used in the evaluation.

Bibliography

Archambault, D., Purchase, H., 2013. Mental map preservation helps user orientation in dynamic graphs. In: Didimo, W., Patrignani, M. (Eds.), Proceedings of the 20th International Symposium on Graph Drawing (GD'12). Vol. 7704 of Lecture Notes in Computer Science. Springer-Verlag, pp. 475–486. Cited on pages 36 and 37.

Archambault, D., Purchase, H. C., Pinaud, B., 2011. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. IEEE Transactions on Visualization and Computer Graphics 17 (4), 539–552. Cited on page 36.

Bender-deMoll, S., McFarland, D. A., 2006. The art and science of dynamic network visualization. Journal of Social Structure 7 (2). Cited on pages 31, 39, and 66.

Bennett, C., Ryall, J., Spalteholz, L., Gooch, A., 2007. The aesthetics of graph visualization. In: Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging. pp. 1–8. Cited on page 9.

Bertin, J., 1983. Semiology of Graphics: Diagrams, Networks, Maps. University of Wisconsin Press. Cited on pages 5, 7, and 9.

Böhringer, K.-F., Paulisch, F. N., 1990. Using constraints to achieve stability in automatic graph layout algorithms. In: Proceedings of the ACM Human Factors in Computing Systems Conference (CHI'90). ACM, The Association for Computing Machinery, pp. 43–51. Cited on pages 33, 41, and 42.

Boitmanis, K., Brandes, U., Pich, C., 2008. Visualizing Internet evolution on the autonomous systems level. In: Proceedings of the 15th International Symposium on Graph Drawing (GD'07). Vol. 4875 of Lecture Notes in Computer Science. Springer-Verlag, pp. 265–276. Cited on pages 40 and 45.

Borg, I., Groenen, P., 2005. Modern Multidimensional Scaling: Theory and Applications, 2nd Edition. Springer Series in Statistics. Springer-Verlag. Cited on page 13.

Borg, I., Lingoes, J., 1980. A model and algorithm for multidimensional scaling with external constraints on the distances. Psychometrika 45 (1), 25–38. Cited on page 28.

Brandenburg, F., Himsolt, M., Rohrer, C., 1996. An experimental comparison of force-directed and randomized graph drawing algorithms. In: Brandenburg, F. (Ed.), Proceedings of the 3rd International Symposium on Graph Drawing (GD'95). Vol. 1027

of Lecture Notes in Computer Science. Springer-Verlag, pp. 76–87. Cited on pages 11 and 13.

Brandes, U., 2001. Drawing on physical analogies. In: Kaufmann, M., Wagner, D. (Eds.), Drawing Graphs: Methods and Models. Vol. 2025 of Lecture Notes in Computer Science. Springer-Verlag, pp. 71–86. Cited on page 10.

Brandes, U., Corman, S. R., 2003. Visual unrolling of network evolution and the analysis of dynamic discourse. Information Visualization 2 (1), 40–50. Cited on pages 39 and 44.

Brandes, U., Indlekofer, N., Mader, M., 2012. Visualization methods for longitudinal social networks and stochastic actor-oriented modeling. Social Networks 34 (3), 291–308. Cited on pages 3 and 61.

Brandes, U., Mader, M., 2012. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In: van Kreveld, M., Speckmann, B. (Eds.), Proceedings of the 19th International Symposium on Graph Drawing (GD'11). Vol. 7034 of Lecture Notes in Computer Science. Springer-Verlag, pp. 99–110. Cited on page 3.

Brandes, U., Pich, C., 2007. Eigensolver methods for progressive multidimensional scaling of large data. In: Kaufmann, M., Wagner, D. (Eds.), Proceedings of the 14th International Symposium on Graph Drawing (GD'06). Vol. 4372 of Lecture Notes in Computer Science. Springer-Verlag, pp. 42–53. Cited on page 26.

Brandes, U., Pich, C., 2009. An experimental study on distance-based graph drawing. In: Proceedings of the 16th International Symposium on Graph Drawing (GD'08). Vol. 5417 of Lecture Notes in Computer Science. Springer-Verlag, pp. 218–229. Cited on pages 2, 14, 23, 24, 25, and 82.

Brandes, U., Pich, C., 2011. More flexible radial layout. Journal of Graph Algorithms and Applications 15 (1), 157–173. Cited on pages 28 and 29.

Brandes, U., Wagner, D., 1997a. A Bayesian paradigm for dynamic graph layout. In: Di Battista, G. (Ed.), Proceedings of the 5th International Symposium on Graph Drawing (GD'97). Vol. 1353 of Lecture Notes in Computer Science. Springer-Verlag, pp. 236–247. Cited on pages 1, 39, 40, 42, 45, and 50.

Brandes, U., Wagner, D., 1997b. Random field models for graph layout. Konstanzer Schriften in Mathematik und Informatik 33. Cited on page 42.

Brandes, U., Wagner, D., 1998. Using graph layout to visualize train interconnection data. In: Whitesides, S. H. (Ed.), Proceedings of the 6th International Symposium on Graph Drawing (GD'98). Vol. 1547 of Lecture Notes in Computer Science. Springer-Verlag, pp. 44–56. Cited on page 11.

Branke, J., 2001. Dynamic graph drawing. In: Kaufmann, M., Wagner, D. (Eds.), Drawing Graphs: Methods and Models. Vol. 2025 of Lecture Notes in Computer Science. Springer-Verlag, pp. 228–246. Cited on pages 33 and 35.

Bridgeman, S. S., Tamassia, R., 2000. Difference metrics for interactive orthogonal graph drawing algorithms. Journal of Graph Algorithms and Applications 4 (3), 47–74. Cited on pages 35, 50, and 80.

Buja, A., Swayne, D. F., 2002. Visualization methodology for multidimensional scaling. Journal of Classification 19 (1), 7–43. Cited on page 23.

Chimani, M., Klein, K., 2010. Algorithm engineering: Concepts and practice. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (Eds.), Experimental Methods for the Analysis of Optimization Algorithms. Springer-Verlag, pp. 131–158. Cited on pages 76 and 77.

Cleveland, W. S., McGill, R., 1984. Graphical perception: Theory, experimentation, and application to the development of graphical methods. Journal of the American Statistical Association 79 (387), 531–554. Cited on page 7.

Cohen, J. D., 1997. Drawing graphs to convey proximity: an incremental arrangement method. ACM Transactions on Computer-Human Interaction 4 (3), 197–229. Cited on pages 15 and 17.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2001. Introduction to Algorithms, 2nd Edition. MIT Press. Cited on page 15.

Cox, T. F., Cox, M. A. A., 2001. Multidimensional Scaling, 2nd Edition. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC. Cited on pages 13, 38, and 87.

Davidson, R., Harel, D., 1996. Drawing graphs nicely using simulated annealing. ACM Transactions on Graphics 15 (4), 301–331. Cited on pages 12 and 13.

de Leeuw, J., 1977. Applications of convex analysis to multidimensional scaling. In: Barra, J. R., Brodeau, F., Romier, G., van Cutsem, B. (Eds.), Recent Developments in Statistics. Amsterdam: North-Holland, pp. 133–145. Cited on page 19.

de Leeuw, J., 1988. Convergence of the majorization method for multidimensional scaling. Journal of Classification 5 (2), 163–180. Cited on pages 19 and 20.

Di Battista, G., Eades, P., Tamassia, R., Tollis, I. G., 1994. Algorithms for drawing graphs: an annotated bibliography. Compututational Geometry: Theory and Appllications 4 (5), 235–282. Cited on page 14.

Di Battista, G., Eades, P., Tamassia, R., Tollis, I. G., 1999. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall. Cited on pages 5, 9, 10, and 35.

Diehl, S., Görg, C., 2002. Graphs, they are changing. In: Proceedings of the 10th International Symposium on Graph Drawing (GD'02). Vol. 2528 of Lecture Notes in Computer Science. Springer-Verlag, pp. 23–30. Cited on pages 39, 43, 51, and 56.

Diehl, S., Görg, C., Kerren, A., 2001. Preserving the mental map using foresighted layout. In: Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization (VisSym '01). Springer-Verlag, pp. 175–184. Cited on pages 43 and 46.

Dijkstra, E. W., 1959. A note on two problems in connection with graphs. Numerische Mathematik 1, 269–271. Cited on page 15.

Dwyer, T., Eades, P., 2002. Visualising a fund manager flow graph with columns and worms. In: Proceedings of the Sixth International Conference on Information Visualisation (IV '02). IEEE Computer Society, pp. 147–152. Cited on page 44.

Dwyer, T., Gallagher, D. R., 2004. Visualising changes in fund manager holdings in two and a half-dimensions. Information Visualization 3 (4), 227–244. Cited on pages 39, 40, and 44.

Dwyer, T., Hong, S.-H., Koschützki, D., Schreiber, F., Xu, K., 2006a. Visual analysis of network centralities. In: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation (APVis'06). pp. 189–197. Cited on pages 40 and 45.

Dwyer, T., Koren, Y., 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 page 28.

Dwyer, T., Koren, Y., Marriott, K., 2006c. Stress majorization with orthogonal ordering constraints. In: Healy, P., Nikolov, N. (Eds.), Proceedings of the 13th International Symposium on Graph Drawing (GD '05). Vol. 3843 of Lecture Notes in Computer Science. Springer-Verlag, pp. 141–152. Cited on pages 28 and 29.

Dwyer, T., Marriott, K., Wybrow, M., 2009. Topology preserving constrained graph layout. In: Proceedings of the 16th International Symposium on Graph Drawing (GD'08). Vol. 5417 of Lecture Notes in Computer Science. Springer-Verlag, pp. 230–241. Cited on page 28.

Eades, P., 1984. A heuristic for graph drawing. Congressus Numerantium 42, 149–160. Cited on pages 10 and 80.

Eades, P., Lai, W., Misue, K., Sugiyama, K., 1991. Preserving the mental map of a diagram. In: Proceedings of Compugraphics '91. pp. 24–33. Cited on pages 1, 31, 33, and 41.

Erten, C., Harding, P., Kobourov, S., Wampler, K., Yee, G., 2004a. Exploring the computing literature using temporal graph visualization. In: Conference on Visualization and Data Analysis (VDA). Vol. 5295 of SPIE proceedings series. pp. 45–56. Cited on page 45.

Erten, C., Harding, P., Kobourov, S., Wampler, K., Yee, G., 2004b. Graphael: Graph animations with evolving layouts. In: Liotta, G. (Ed.), Proceedings of the 11th International Symposium on Graph Drawing (GD'03). Vol. 2912 of Lecture Notes in Computer Science. Springer-Verlag, pp. 98–110. Cited on page 45.

Erten, C., Kobourov, S. G., Le, V., Navabi, A., 2004c. Simultaneous Graph Drawing: Layout Algorithms and Visualization Schemes. In: Proceedings of the 11th International Symposium on Graph Drawing (GD'03). Vol. 2912 of Lecture Notes in Computer Science. Springer-Verlag, pp. 437–449. Cited on pages 40, 44, 45, and 46.

Farrugia, M., Quigley, A., 2011. Effective temporal graph layout: A comparative study of animation versus static display methods. Information Visualization 10 (1), 47–64. Cited on page 36.

Fisher, R. A., 1935. The design of experiments. Oliver and Boyd, Edinburgh. Cited on page 77.

Floyd, R. W., 1962. Algorithm 97: Shortest path. Communications of the ACM 5 (6), 345. Cited on page 15.

Fredman, M. L., Tarjan, R. E., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34 (3), 596–615. Cited on page 15.

Freeman, L. C., 2000. Visualizing social networks. Journal of Social Structure 1 (1). Cited on page 1.

Frick, A., Ludwig, A., Mehldau, H., 1995. A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In: Tamassia, R., Tollis, I. (Eds.), Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94). Vol. 894 of Lecture Notes in Computer Science. Springer-Verlag, pp. 388–403. Cited on pages 11 and 12.

Friedrich, C., Eades, P., 2002. Graph drawing in motion. Journal of Graph Algorithms and Applications 6 (3), 353–370. Cited on page 37.

Friedrich, C., Houle, M. E., 2002. Graph drawing in motion II. In: Proceedings of the 9 International Symposium on Graph Drawing (GD'01). Vol. 2265 of Lecture Notes in Computer Science. Springer-Verlag, pp. 122–125. Cited on page 37.

Frishman, Y., Tal, A., 2008. Online dynamic graph drawing. IEEE Transactions on Visualization and Computer Graphics 14 (4), 727–740. Cited on pages 40, 45, and 81.

Fruchterman, T. M., Reingold, E. M., 1991. Graph-drawing by force-directed placement. Software—Practice and Experience 21 (11), 1129–1164. Cited on pages 12 and 39.

Gajer, P., Goodrich, M. T., Kobourov, S. G., 2004. A multi-dimensional approach to force-directed layouts of large graphs. Computational Geometry 29 (1), 3–18. Cited on pages 12 and 25.

Gansner, E., Hu, Y., North, S., 2013. A maxent-stress model for graph layout. IEEE Transactions on Visualization and Computer Graphics 19 (6), 927–940. Cited on page 27.

Gansner, E., Koren, Y., North, S., 2004. Graph drawing by stress majorization. In: Proceedings of the 12th International Symposium on Graph Drawing (GD'04). Vol. 3383 of Lecture Notes in Computer Science. Springer-Verlag, pp. 239–250. Cited on pages 14, 19, and 21.

Gansner, E. R., Hu, Y., 2009. Efficient node overlap removal using a proximity stress model. In: Proceedings of the 16th International Symposium on Graph Drawing (GD'08). Vol. 5417 of Lecture Notes in Computer Science. Springer-Verlag, pp. 206–217. Cited on pages 28 and 29.

Gansner, E. R., Hu, Y., North, S., Scheidegger, C., 2011. Multilevel agglomerative edge bundling for visualizing large graphs. In: 2011 IEEE Pacific Visualization Symposium. pp. 187–194. Cited on page 26.

Geipel, M. M., 2007. Self-organization applied to dynamic network layout. International Journal of Modern Physics C 18 (10), 1537–1549. Cited on page 39.

Gilbert, E. N., 1959. Random graphs. The Annals of Mathematical Statistics 30 (4), 1141–1144. Cited on page 85.

Golub, G. H., Van Loan, C. F., 1996. Matrix Computations, 3rd Edition. John Hopkins University Press. Cited on page 26.

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 22 and 24.

Griffin, A. L., MacEachren, A. M., Hardisty, F., Steiner, E., Li, B., 2006. A comparison of animated maps with static small-multiple maps for visually identifying space-time clusters. Annals of the Association of American Geographers 96 (4), 740–753. Cited on page 36.

Groenen, P., Heiser, W., 1996. The tunneling method for global optimization in multidimensional scaling. Psychometrika 61 (3), 529–550. Cited on page 22.

Groenen, P. J. F., Heiser, W. J., Meulman, J. J., 1999. Global optimization in least-squares multidimensional scaling by distance smoothing. Journal of Classification 16 (2), 225–254. Cited on page 22.

Groh, G., Hanstein, H., Wörndl, W., 2009. Interactively visualizing dynamic social networks with DySoN. In: Proceedings of the IUI'09 Workshop on Visual Interfaces to the Social and Semantic Web (VISSW'09). Vol. 443. CEUR Workshop Proceedings. Cited on page 39.

Hachul, S., Jünger, M., 2004. Drawing large graphs with a potential-field-based multilevel algorithm. In: Proceedings of the 12th International Symposium on Graph Drawing (GD'04). Vol. 3383 of Lecture Notes in Computer Science. Springer-Verlag, pp. 285–295. Cited on pages 12 and 25.

Hadany, R., Harel, D., 2001. A multi-scale algorithm for drawing graphs nicely. Discrete Applied Mathematics 113 (1), 3–21. Cited on page 12.

Harel, D., Koren, Y., 2002. A fast multi-scale method for drawing large graphs. Journal of Graph Algorithms and Applications 6 (3), 179–202. Cited on page 12.

Herman, I., Melançon, G., Marshall, M. S., 2000. Graph visualization and navigation in information visualization: a survey. IEEE Transactions on Visualization and Computer Graphics 6 (1), 24–43. Cited on page 5.

Himsolt, M., 1995. Comparing and evaluating layout algorithms within GraphEd. Journal of Visual Languages and Computing 6 (3), 255–273. Cited on page 80.

Huang, M., Eades, P., 1998. A fully animated interactive system for clustering and navigating huge graphs. In: Whitesides, S. (Ed.), Proceedings of the 6th International Symposium on Graph Drawing (GD'98)Graph Drawing. Vol. 1547 of Lecture Notes in Computer Science. Springer-Verlag, pp. 374–383. Cited on page 11.

Huang, M. L., Eades, P., Wang, J., 1998. On-line animated visualization of huge graphs using a modified spring algorithm. Journal of Visual Languages and Computing 9 (6), 623–645. Cited on page 39.

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., Morris, M., 2008. ergm: A package to fit, simulate and diagnose exponential-family models for networks. Journal of Statistical Software 24 (3). Cited on page 83.

Jaromczyk, J., Toussaint, G., 1992. Relative neighborhood graphs and their relatives. Proceedings of the IEEE 80, 1502–1517. Cited on page 35.

Johnson, D. S., 1982. The $\mathcal{NP}$-completeness column: An ongoing guide. Journal of Algorithms 3 (1), 89–99. Cited on page 10.

Johnson, D. S., 2002. A theoretician's guide to the experimental analysis of algorithms. Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges 59, 215–250. Cited on page 76.

Kamada, T., Kawai, S., 1988. A simple method for computing general positions in displaying three-dimensional objects. Computer Vision, Graphics and Image Processing 41, 43–56. Cited on pages 11, 13, 15, 17, and 39.

Kaufmann, M., Wagner, D. (Eds.), 2001. Drawing Graphs: Methods and Models. Vol. 2025 of Lecture Notes in Computer Science. Springer-Verlag. Cited on pages 5 and 9.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680. Cited on page 12.

Klimenta, M., 2012. Extending the usability of multidimensional scaling for graph drawing. Ph.D. thesis, Universität Konstanz. Cited on pages 13 and 26.

Klovdahl, A. S., 1981. A note on images of networks. Social Networks 3, 197–214. Cited on page 1.

Knecht, A. B., 2008. Friendship selection and friends' influence. dynamics of networks and actor attributes in early adolescence. Ph.D. thesis, Utrecht University. Cited on page 1.

Koffka, K., 1935. Principles of Gestalt psychology. New York: Harcourt, Brace. Cited on page 9.

Koren, Y., Çivril, A., 2009. The binary stress model for graph drawing. In: Tollis, I., Patrignani, M. (Eds.), Proceedings of the 16th International Symposium on Graph Drawing (GD '08). Vol. 5417 of Lecture Notes in Computer Science. Springer-Verlag, pp. 193–205. Cited on pages 28 and 29.

Kruskal, J. B., March 1964a. Multidimensional scaling by optimizing goodness of fit to a nonparametric hypothesis. Psychometrika 29, 1–27. Cited on page 17.

Kruskal, J. B., June 1964b. Nonmetric multidimensional scaling: A numerical method. Psychometrika 29, 115–129. Cited on pages 19 and 22.

Kruskal, J. B., 1971. Comments on "a nonlinear mapping for data structure analysis". IEEE Transactions on Computers C-20 (12), 1614. Cited on page 17.

Kruskal, J. B., Seery, J. B., 1980. Designing network diagrams. In: Proceedings of the First General Conference on Social Graphics. pp. 22–50. Cited on page 14.

Kruskal, J. B., Wish, M., 1978. Multidimensional Scaling. Vol. 07–011 of Sage University Paper series on Quantitative Applications in the Social Sciences. Sage Publications. Cited on page 13.

Lazega, E., Lemercier, C., Mounier, L., 2006. A spinning top model of formal organization and informal behavior: dynamics of advice networks among judges in a commercial court. European Management Review 3 (2), 113–122. Cited on page 1.

Leydesdorff, L., Schank, T., 2008. Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. Journal of the American Society for Information Science and Technology 59 (11), 1810–1818. Cited on pages 40 and 45.

Lipp, C., 2000. Political and revolutionary culture in a german town 1830-1850: A prosopographical approach. History and Computing 12 (1), 73–83. Cited on page 17.

Lyons, K. A., 1992. Cluster busting in anchored graph drawing. In: Proceedings of the 1992 CAS Conference (CASCON '92). IBM Press, pp. 7–17. Cited on pages 41, 43, and 45.

Lyons, K. A., Meijer, H., Rappaport, D., 1998. Algorithms for cluster busting in anchored graph drawing. Journal of Graph Algorithms and Applications 2 (1), 1–24. Cited on pages 40, 42, and 81.

Manger, M. S., Pickup, M. A., Snijders, T. A. B., 2012. A hierarchy of preferences. Journal of Conflict Resolution 56 (5), 853–878. Cited on page 1.

Mardia, K., 1978. Some properties of clasical multi-dimesional scaling. Communications in Statistics - Theory and Methods 7 (13), 1233–1241. Cited on page 24.

McGee, V. E., 1966. The multidimensional analysis of 'elastic' distances. British Journal of Mathematical and Statistical Psychology 19 (2), 181–196. Cited on page 17.

Michell, L., Amos, A., 1997. Girls, pecking order and smoking. Social Science & Medicine 44 (12), 1861–1869. Cited on page 85.

Misue, K., Eades, P., Lai, W., Sugiyama, K., 1995. Layout adjustment and the mental map. Journal on Visual Languages and Computing 6 (2), 183–210. Cited on page 41.

Montgomery, D. C., 2013. Design and Analysis of Experiments, 8th Edition. Wiley. Cited on pages 77, 78, and 81.

Moody, J., McFarland, D. A., Bender-deMoll, S., 2005. Dynamic network visualization. American Journal of Sociology 110 (4), 1206–1241. Cited on pages 39 and 46.

Moreno, J. L., 1953. Who Shall Survive: Foundations of Sociometry, Group Psychotherapy, and Sociodrama. Beacon House. Cited on page 8.

Moret, B. M., 2002. Towards a discipline of experimental algorithmics. Data Structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges 59, 197–213. Cited on pages 75, 81, and 83.

Nesbitt, K. V., Friedrich, C., 2002. Applying gestalt principles to animated visualizations of network data. In: Proceedings of the 6th International Conference on Information Visualisation (IV'02). IEEE Press, pp. 337–343. Cited on page 37.

North, S. C., 1996. Incremental layout with DynaDag. In: Proceedings of the 3rd International Symposium on Graph Drawing (GD'95). Vol. 1027 of Lecture Notes in Computer Science. Springer-Verlag, pp. 409–418. Cited on page 42.

Pich, C., 2009. Applications of multidimensional scaling to graph drawing. Ph.D. thesis, Universität Konstanz. Cited on pages 13 and 27.

Popper, K. R., 1934. Logik der Forschung. Springer-Verlag. Cited on page 76.

Purchase, H. C., 1998. Performance of layout algorithms: Comprehension, not computation. Journal of Visual Languages & Computing 9 (6), 647–657. Cited on page 80.

Purchase, H. C., Cohen, R. F., James, M., 1997. An experimental study of the basis for graph drawing algorithms. ACM Journal of Experimental Algorithmics 2 (4), 1–17. Cited on pages 9 and 80.

Purchase, H. C., Samra, A., 2008. Extremes are better: Investigating mental map preservation in dynamic graphs. In: Proceedings of the 5th International Conference on Diagrammatic Representation and Inference (Diagrams 2008). Vol. 5223 of Lecture Notes in Computer Science. Springer-Verlag, pp. 60–73. Cited on pages 35, 80, and 97.

Quinn, N. R., 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 10.

Rardin, R. L., Uzsoy, R., 2001. Experimental evaluation of heuristic optimization algorithms: A tutorial. Journal of Heuristics 7 (3), 261–304. Cited on page 77.

Robertson, G., Fernandez, R., Fisher, D., Lee, B., Stasko, J., 2008. Effectiveness of animation in trend visualization. IEEE Transactions on Visualization and Computer Graphics 14 (6), 1325–1332. Cited on page 36.

Robins, G., Pattison, P., Kalish, Y., Lusher, D., 2007. An introduction to exponential random graph (p*) models for social networks. Social Networks 29 (2), 173–191. Cited on page 83.

Saffrey, P., Purchase, H., 2008. The "mental map" versus "static aesthetic" compromise in dynamic graphs: A user study. In: Proceedings of the 9th Australasian User Interface Conference (AUIC 2008). pp. 85–93. Cited on pages 35 and 87.

Sammon, J., 1969. A nonlinear mapping for data structure analysis. IEEE Transactions on Computers 18 (5), 401–409. Cited on page 17.

Sanders, P., 2009. Algorithm Engineering – an attempt at a definition. In: Efficient Algorithms. Vol. 5760 of Lecture Notes in Computer Science. Springer-Verlag, pp. 321–340. Cited on pages 76 and 82.

Sibson, R., 1978. Studies in the robustness of multidimensional scaling: Procrustes statistics. Journal of the Royal Statistical Society. Series B (Methodological) 40 (2), 234–238. Cited on pages 24 and 38.

Snijders, T. A. B., 2001. The statistical evaluation of social network dynamics. Sociological Methodology 31, 361–395. Cited on page 83.

Snijders, T. A. B., Koskinen, J., Schweinberger, M., 2010. Maximum likelihood estimation for social network dynamics. Annals of Applied Statistics 4 (2), 567–588. Cited on page 83.

Sugiyama, K., Misue, K., 1995. A simple and unified method for drawing graphs: Magnetic-spring algorithm. In: Tamassia, R., Tollis, I. (Eds.), DIMACS International Workshop on Graph Drawing (GD'94). Vol. 894 of Lecture Notes in Computer Science. Springer-Verlag, pp. 364–375. Cited on page 11.

Sugiyama, K., Tagawa, S., Toda, M., 1981. Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man and Cybernetics 11, 109–125. Cited on page 41.

Taylor, M., Rodgers, P., 2005. Applying graphical design techniques to graph visualisation. In: Proceedings of the Ninth International Conference on Information Visualisation. IEEE Computer Society, pp. 651–656. Cited on page 9.

Tichy, W., 1998. Should computer scientists experiment more? Computer 31 (5), 32–40. Cited on page 75.

Torgerson, W. S., 1952. Multidimensional scaling: I. Theory and Method. Psychometrika 17, 401–419. Cited on page 22.

Tufte, E. R., 1983. The Visual Display of Quantitative Information. Graphics Press. Cited on page 9.

Tufte, E. R., 1990. Envisioning Information. Graphics Press. Cited on page 36.

Tversky, B., Bauer Morrison, J., Betrancourt, M., 2002. Animation: can it facilitate? International Journal of Human-Computer Studies 57 (4), 247–262. Cited on page 36.

Van De Bunt, G. G., Van Duijn, M. A., Snijders, T. A., 1999. Friendship networks through time: An actor-oriented dynamic statistical network model. Computational & Mathematical Organization Theory 5 (2), 167–192. Cited on page 85.

Vismara, L., Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Vargiu, F., 2000. Experimental studies on graph drawing algorithms. Software – Practice and Experperience 30 (11), 1235–1284. Cited on page 80.

Walshaw, C., 2001. A multilevel algorithm for force-directed graph drawing. In: Marks, J. (Ed.), Proceedings of the 8th International Symposium on Graph Drawing (GD 2000). Vol. 1984 of Lecture Notes in Computer Science. Springer-Verlag, pp. 171–182. Cited on page 12.

Ware, C., Purchase, H., Colpoys, L., McGill, M., 2002. Cognitive measurements of graph aesthetics. Information Visualization 1 (2), 103–110. Cited on page 9.

Warshall, S., 1962. A theorem on boolean matrices. Journal of the ACM 9 (1), 11–12. Cited on page 15.

Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H., 2012. Interactive level-of-detail rendering of large graphs. IEEE Transactions on Visualization and Computer Graphics 18 (12), 2486–2495. Cited on page 26.