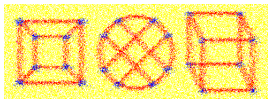


Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 11, no. 2, pp. 325–343 (2007)

Dynamic Spectral Layout with an Application to Small Worlds

Ulrik Brandes *Daniel Fleischer* *Thomas Puppe*

Department of Computer & Information Science
University of Konstanz, Germany
{Ulrik.Brandes,Daniel.Fleischer,Thomas.Puppe}@uni-konstanz.de

Abstract

Spectral methods are naturally suited for dynamic graph layout because, usually, moderate changes of a graph yield moderate changes of the layout. We discuss some general principles for dynamic graph layout and derive a dynamic spectral layout approach for the animation of small-world models.

Article Type	Communicated by	Submitted	Revised
Regular paper	P. Eades and P. Healy	January 2006	January 2007

1 Introduction

The main problem in dynamic graph layout is the balance of layout quality and mental-map preservation [17]. Typically, the problem is addressed by adapting a static layout method such that it produces similar layouts for successive graphs. While these adaptations are typically ad-hoc [8], others [2, 1] are based on the formally derived method [3] of integrating difference metrics [5] into the static method. See [4] for an overview of the dynamic graph drawing problem.

Spectral layout denotes the use of eigenvectors of graph-related matrices such as adjacency or Laplacian matrix as coordinate vectors. See, e.g., [15] for an introduction. We argue that spectral methods are naturally suited for dynamic graph layout both from a theoretical and practical point of view. Usually, moderate changes in the graph yield moderate changes of the layout, such that updates of spectral layouts can be computed efficiently. This holds in particular for small worlds (see Sect. 5), where the initial ring structure provides stability of the layout, see Fig. 9.

This paper is organized as follows. In Sect. 2, we define some basic notation and recall principles of spectral graph layout. The dynamic graph layout problem is reviewed briefly in Sect. 3, and methods for updates between layouts of consecutive graphs are treated in more detail in Sect. 4. In Sect. 5, our approach for small worlds is introduced, and we conclude with a brief discussion in Sect. 6.

2 Preliminaries

For ease of exposition we consider only two-dimensional straight-line representations of simple, undirected graphs $G = (V, E)$ with positive edge weights $\omega\colon E \to \mathbb{R}^+$, although most techniques and results in this paper easily carry over to other classes of graphs.

In straight-line representations, a two-dimensional *layout* is determined by a vector $\big(p(v)\big)_{v\in V}$ of *positions* $p(v) = \big(x(v), y(v)\big)$. Most of the time we will reason about one-dimensional layouts x that represent the projection of p onto one component.

For any graph-related matrix $M(G)$, a (two-dimensional) *spectral layout* of G is defined by two eigenvectors x and y of $M(G)$. For simplicity, we will only consider layouts derived from the *Laplacian matrix* $L = L(G)$ of G, which is defined by elements

$$L_{v,w} := \begin{cases} \sum_{u\in V} \omega(u,v) & \text{if } v = w\ , \\ -\omega(v,w) & \text{if } v \neq w\ , \\ 0 & \text{otherwise.} \end{cases}$$

The rows of $L(G)$ add up to 0, thus, the vector $\mathbf{1} := (1, \ldots, 1)^{\mathrm{T}}$ is a trivial eigenvector for eigenvalue 0. Since $L(G)$ is symmetric, all eigenvalues are real, and the theorem of Gershgorin [13] yields that they are bounded into $[0, g]$,

where g is two times the maximum (weighted) degree of G. Hence, the spectrum can be written as $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq g$ with corresponding unit eigenvectors $\mathbf{1}/\sqrt{n} = u_1, \ldots, u_n$.

Based on the Laplacian, a spectral layout is defined as $p = (u_2, u_3)$, where u_2 and u_3 are unit eigenvectors to the second and third smallest eigenvalues λ_2, λ_3 of the corresponding Laplacian matrix L. This has already been used for graph drawing in 1970 by Hall [14].

For sparse graphs of moderate size, a practical method to determine the corresponding eigenvectors is *power iteration*. For an initial vector x the matrix-vector multiplication $Lx/\|Lx\|$ is iterated until it converges to a unit eigenvector associated with the largest eigenvalue. Since we are not interested in u_n, we use matrix $\hat{L} = g \cdot I - L$, which has the same eigenvectors with the order of their eigenvalues $g = g - \lambda_1 \geq g - \lambda_2 \geq \ldots \geq g - \lambda_n$ reversed. To obtain u_2 and u_3, respectively, x is orthogonalized with u_1 (and in the case of u_3 also with u_2) after each iteration step, i.e., the mean value $\sum_{v \in V} x(v)/n$ is subtracted from every element of x to ensure $x \perp \mathbf{1}$. Spectral layouts of larger graphs can be computed efficiently using multiscale methods [16].

3 Dynamic Layout

In our setting, a *dynamic graph* is a sequence $G^{(0)}, G^{(1)}, \ldots, G^{(t_{\max})}$ of graphs with, usually, small edit distance, i.e., $G^{(t+1)}$ is obtained from $G^{(t)}$ by adding, changing, and deleting only a few vertices and edges.

There are two main scenarios for the animation of a dynamic graph, depending on whether the individual graphs are presented to the layout algorithm one at a time, or the entire sequence is known in advance. Layout approaches for the *offline* scenario (e.g., [7]) are frequently based on a layout of the union of all graphs in the sequence. A variant are 2.5D representations in which all graphs are shown at once (e.g., [9]). In the *online* scenario, the typical approach is to consider only the previous layout (e.g., [8]). A variant in which provisions for likely future changes are made is presented in [6].

We here focus on the online scenario. In fact, we even restrict our attention to the elementary case of an update step between two consecutive layouts.

4 Updates

Assume we are given a sequence of layouts $p_0, p_1, \ldots, p_{t_{\max}}$ for a dynamic graph $G^{(0)}, G^{(1)} \ldots, G^{(t_{\max})}$. The step from p_t to p_{t+1} is called the *logical update*, whereas the actual animation of the transition is referred to as the *physical update*.

While simple, say, *linear interpolation* of two layouts is most frequently used in graph editors, more sophisticated techniques for morphing are available (see, e.g., [11, 12, 10]). General morphing strategies do not take into account the method by which origin and target layout are generated.

For dynamic spectral layout, at least two additional strategies are reasonable.

4.1 Iteration

We focus w.l.o.g. on the step from $G^{(0)}$ to $G^{(1)}$. If the target layout x_1 is a spectral layout, the power iteration for its own computation can and should be initialized with x_0, that will usually be close to the target layout. The power iteration then produces intermediate layouts which can be used for the physical update. A way to enhance the smoothness of morphing is needed because of the observation, that the first steps of the iteration yield greater movement of the vertices when compared to later steps. Let $\hat{L} = g \cdot I - L(G^{(1)})$. An iteration step consists of computing the new layout $\hat{L}x/\|\hat{L}x\|$ from a given layout $x \perp \mathbf{1}$. Let $g = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and $\mathbf{1}/\sqrt{n} = u_1, u_2, \ldots, u_n$ be the eigenvalues and unit eigenvectors of $\hat{L}$, respectively. Then if $\lambda_1 > \lambda_2 > \lambda_3$ (otherwise just proper eigenvectors and eigenvalues would have to be chosen in what follows) and $x = \sum_{j=2}^{n} \alpha_j u_j, \alpha_2 \neq 0$ we have

$$\begin{aligned}
\frac{\hat{L}^k x}{\|\hat{L}^k x\|} &\longrightarrow u_2 \quad \text{and} \\
\Big|\Big| u_2 - \frac{\hat{L}^k x}{\|\hat{L}^k x\|} \Big|\Big| &= \Big|\Big| u_2 - \frac{\sum_{j=2}^{n} \lambda_j^k \alpha_j u_j}{\|\hat{L}^k x\|} \Big|\Big| \\
&\leq 1 - \frac{\lambda_2^k \alpha_2}{\|\hat{L}^k x\|} + \frac{\sum_{j=3}^{n} \lambda_j^k \alpha_j}{\|\hat{L}^k x\|} \\
&= \mathcal{O}\big((\lambda_3/\lambda_2)^k\big) \; .
\end{aligned}$$

One way to handle this non-linear decay is to use layouts after appropriately spaced numbers of steps, or to use layouts only if the difference to the last used layout exceeds some threshold c in some metric, e.g., if $\|x - x'\| > c$. Both ways will enhance the smoothness of morphing by avoiding the drawing of many small movements at the end of the iteration process. Finally, the overall number of iterations can always be used to choose between layout quality and mental-map preservation, where a small number of iterations favors the latter case.

4.2 Interpolation

If both origin and target layout x_0 and x_1 are spectral layouts, intermediate layouts can also be obtained by computing eigenvectors of some intermediate matrices from $L(G^{(0)})$ to $L(G^{(1)})$. We interpolate linearly by

$$(t-1)L(G^{(0)}) + tL(G^{(1)}), \quad 0 \leq t \leq 1 \; .$$

Layouts are computed for a sequence of *breakpoints* (each corresponds to a frame of the animation) $0 \leq t_1 < t_2 < \cdots < t_k \leq 1$ ($t_{j+1} - t_j$ constant, or proportional to $\sin(\pi j/k)$, depending on what kind of morphing seems to be appropriate, the latter one slowing down at the beginning and end). For every breakpoint t_j the

iteration is initialized with the layout of t_{j-1}, which allows fast convergence and small movements between two succeeding breakpoints. Deletion and insertion of vertices have to be handled in a different manner since the matrix dimension changes. See Sect. 5 for details.

Figs. 6 and 8 show smooth animations of this method. Theoretical justification for smoothness comes along with a theorem by Rellich [18] applied to the finite dimensional case. Matrix $(1-t)L(G^{(0)}) + tL(G^{(1)})$ can be seen as a perturbed self-adjoint operator $L(t) = L(G^{(0)}) + t\big(L(G^{(1)}) - L(G^{(0)})\big)$ with corresponding eigenvalues $\lambda_j(t)$ and eigenvectors $u_j(t)$, that are holomorphic with respect to t, i.e.,

$$L(t)u_j(t) = \lambda_j(t)u_j(t), \tag{1}$$

where $u_j(0)$ are eigenvectors at time $t = 0$ and $u_j(1)$ can be permuted by a permutation π, such that $u_{\pi(j)}(1)$ are (ordered) eigenvectors at time $t = 1$. Note that two eigenvectors may only have to be exchanged if their corresponding eigenvalues intersect during the time from 0 to 1. And even then, the power iteration exchanges these eigenvectors sufficiently smoothly for pleasing animations because the corresponding eigenvalues remain within the same range for some time.

Consider λ_2 and u_2 of a small world with n vertices, where starting from a circle, each vertex is connected to its $2k$ nearest neighbors, and $p = 0$, see Sect. 5 about the small world model. Both $\lambda_2(t)$ and $u_2(t)$ can locally be written as power series

$$\begin{aligned} \lambda_2(t) &= \mu_0 + t\mu_1 + t^2\mu_2 + \cdots \ , \\ u_2(t) &= w_0 + tw_1 + t^2w_2 + \cdots \ . \end{aligned} \tag{2}$$

We call $\lambda_2(t)$ and $u_2(t)$ *smooth functions* if $\|w_j\| \le 1$ and $|\mu_j| \le 8/\sqrt{n}$ for $j > 0$. Clearly, this condition yields a smooth animation, say, within $[0, 1/2]$. Note that the condition is very strict and not necessary for a smooth animation. In fact, all animations when inserting an edge into a small world with arbitrary parameters n, k and $p = 0$ can be called smooth since the inserted edge just narrows the original circle. Nevertheless, this strict condition can be seen to be preserved also for small $p \neq 0$ in practice, see Figs. 2 to 4, implying that animations with these graphs are also smooth.

Theorem 1 *The eigenvalue $\lambda_2(t)$ and the eigenvector $u_2(t)$ when inserting an edge into a small world with $\sqrt{n} \le k \le n/5$ (and $p = 0$) are smooth functions.*

Proof: We write $L(t) = L + tP$, where P is the insertion of an edge between two non-adjacent vertices (with indices 1 and r), and denote by I the identity matrix. From (1) and (2) we get

$$\begin{aligned} Lw_0 &= \mu_0 w_0 \ , \\ (L - \mu_0 I)w_j &= \sum_{\ell=0}^{j-1} \mu_{j-\ell} w_i - Pw_{j-1} \ , (j > 0) \ , \end{aligned} \tag{3}$$

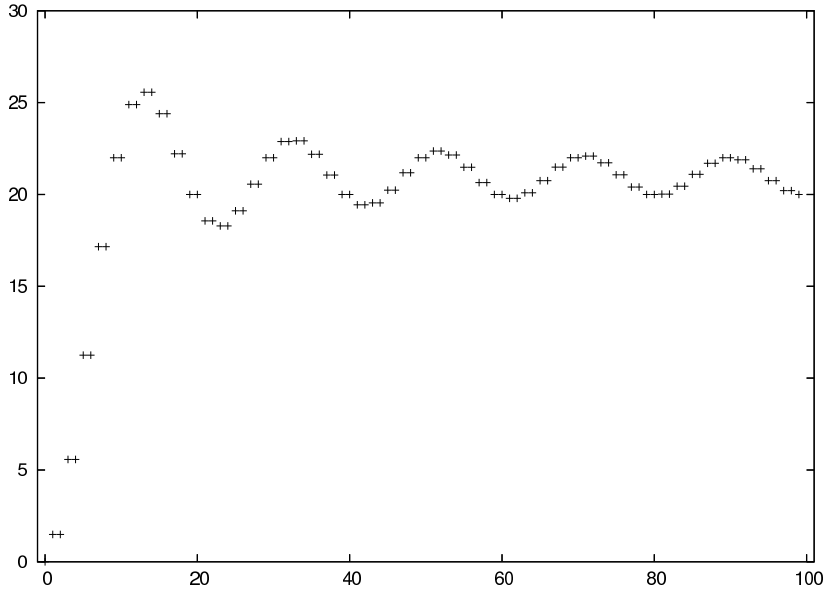


Figure 1: The eigenvalues λ_i of a small world with $n = 100, k = 10, p = 0$

where we can recursively choose $w_j, (j > 0)$ such that $w_j \perp w_0$. Since the right hand sides of (3) need to be orthogonal to w_0 for $j > 0$, this yields

$$\mu_j = \langle P w_{j-1}, w_0 \rangle \ , (j > 0) \ .$$

A set of unit eigenvectors is, e.g.,

$$\begin{aligned} u_1 &= (1, \ldots, 1)^{\mathrm{T}}/\sqrt{n} \ , \\ u_j &= (n/2)^{-1/2} \big(\cos(\lfloor j/2 \rfloor 2\pi\ell/n + (-1)^j \pi/4) \big)^{\mathrm{T}}_{\ell=1,\ldots,n}, (j > 1) \ , \end{aligned}$$

and the corresponding eigenvalues are

$$\lambda_j = 2k - 2 \sum_{\ell=1}^{k} \cos(\lfloor j/2 \rfloor 2\pi\ell/n) \ .$$

Note that in this order these are not monotone, see Fig. 1. The multiplicity of λ_2 is 2, and the right hand sides of (3) are also orthogonal to u_1, such that $1/(\lambda_4 - \lambda_2)$ is the least upper bound of the inverse mapping of $L - \mu_0 I$ applied to the right hand side. Now we can recursively compute bounds for $|\mu_j|, \|w_j\|$.

$$\begin{aligned} |\mu_1| &= (w_{0,1} - w_{0,r})^2 \leq 8/n \leq 8/\sqrt{n} \ , \\ \|w_1\| &\leq \frac{\sqrt{2}|w_{0,1} - w_{0,r}|}{\lambda_4 - \lambda_2} \leq \frac{4/\sqrt{n}}{\lambda_4 - \lambda_2} =: q \ , \\ |\mu_j| &\leq C_{j-1} q^{j-1} 4/\sqrt{n} \ , \\ \|w_j\| &\leq C_j q^j \ , \end{aligned}$$

where C_j stems from the convolution in (3) and is defined by $C_0 = 1$ and $C_{j+1} = \sum_{\ell=0}^{j} C_\ell C_{j-\ell}$ for $j \geq 0$. Since Lemma 1 states that $C_j \leq 4^j$, it remains to be shown that $q \leq 1/4$.

$$\begin{aligned}
\lambda_4 - \lambda_2 &\geq k - \sum_{\ell=1}^{k} \cos(4\pi\ell/n) \\
&= (k+1) - \frac{\cos(2k\pi/n)\sin(2(k+1)\pi/n)}{\sin(2\pi/n)} \\
&\geq (k+1)\Big(\frac{1}{2}(2k\pi/n)^2 - \frac{1}{24}(2k\pi/n)^4\Big) \\
&\geq \frac{k^3}{2}(2\pi/n)^2\Big(1 - \frac{1}{12}(2\pi/5)^2\Big) \ .
\end{aligned}$$

Together with $k \geq \sqrt{n}$ this yields $q \leq 1/4$. □

The bounds for k can be improved when we do not use the least upper bound of the inverse mapping of $L - \mu_0 I$, but we assume that the right hand sides are independent random vectors and consider expected values. The lower bound for k then becomes proportional to the cubic root of n. Anyhow, since during the summation in (3) many vectors may cancel out, Theorem 1 is not tight. In fact, as can be seen in Fig. 2, $\lambda_2(t)$ is a smooth function also in some range outside these bounds.

Figs. 2 to 4 also show that the property of smooth functions carries over to small worlds with $p \neq 0$ to some extent. While it still holds quite well for small worlds up to $p = 0.05$, it is clearly no longer fulfilled for $p = 0.4$.

Lemma 1 *$C_j \leq 4^j$ for all $j \geq 0$.*

Proof: The sequence C_j is well-known as Catalan numbers whose generating function $(1 - \sqrt{1 - 4x})/2x$ delivers

$$C_j = \frac{1}{j+1}\binom{2j}{j} \ .$$

Now, $C_0 = 1 \leq 4^0$ and $C_{j+1}/C_j = 4(j + 1/2)/(j + 2) < 4$ proves the lemma. □

5 Application to Small Worlds

Watts and Strogatz [19] introduced a random graph model that captures some often-observed features of empirical graphs simultaneously: sparseness, local clustering, and small average distances. This is achieved by starting from a cycle and connecting each node with its $2k$ nearest neighbors for some small, fixed k. The resulting graph is sparse and has a high clustering coefficient (average density of vertex neighborhoods), but also high (linear) average distance.

The average distance drops quickly when only a few random edges are rewired randomly. If each edge is rewired independently with some probability p, there is a large interval of p in which the average distance is already logarithmic while the clustering coefficient is still reasonably high.

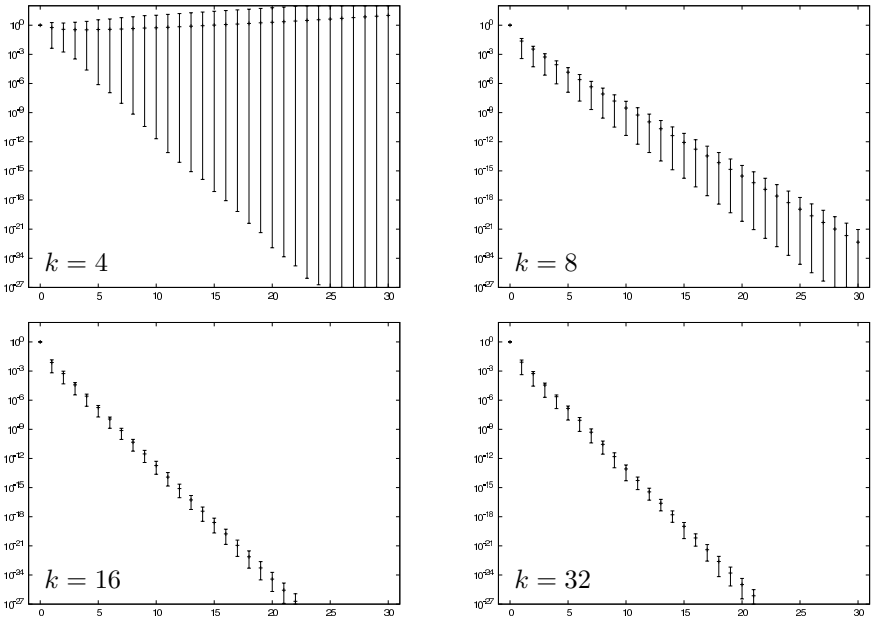


Figure 2: Maximum, minimum and average norm of the coefficients of $u_2(t)$ of 100 samples of a small world with $n = 100, p = 0$ and $k = 4, 8, 16, 32$

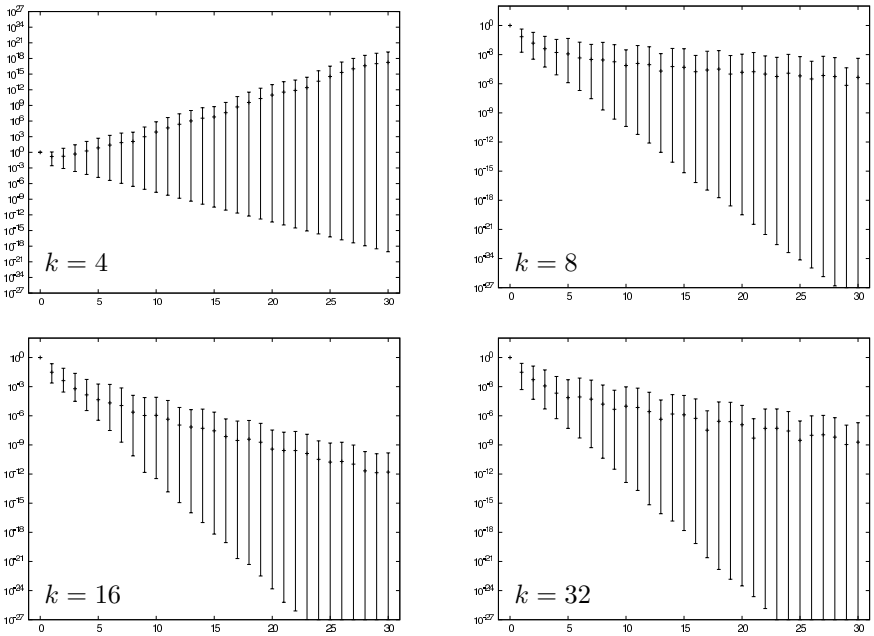


Figure 3: Maximum, minimum and average norm of the coefficients of $u_2(t)$ of 100 samples of a small world with $n = 100, p = 0.05$ and $k = 4, 8, 16, 32$

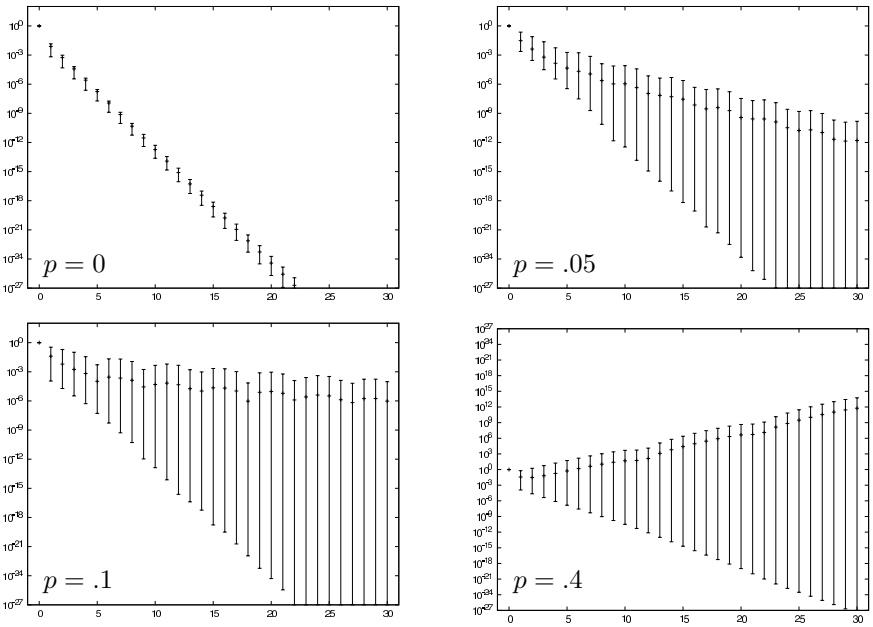


Figure 4: Maximum, minimum and average norm of the coefficients of $u_2(t)$ of 100 samples of a small world with $n = 100, k = 16$ and $p = 0, .05, .1, .4$

5.1 Dynamic Laplacian layout

Interestingly, spectral layouts highlight the construction underlying the above model and thus point to the artificiality of generated graphs. This is due to the fact that spectral layouts of regular structures display their symmetry very well, and are, usually, only moderately disturbed by small perturbations in the graph (mirroring the argument for their use in dynamic layout). The initial ring structure of the small world in Fig. 9 is still apparent, even though a significant number of chords have been introduced by random rewiring. In fact, the layout conveys very well which parts of the ring were joined by short-cut edges.

Figs. 5 and 6 point out differences between the two approaches using intermediate layouts obtained from the power iteration and from matrix interpolation. It can be seen that the power iteration first acts locally around the changes. This stems from the fact that in the first multiplication only the neighborhood of the change, i.e., the two incident vertices of an edge with changed weight or the neighbors of a deleted or inserted vertex, are affected. The next step also affects vertices at distance 2, and so on. Hence, the change spreads like a wavefront. The matrix interpolation approach acts globally at every step. Interpolating the Laplacian matrices corresponds to gradually changing edge weights. The animation therefore is much more smooth.

Figs. 7 and 8 show differences between simple linear interpolation of the positions and matrix interpolation. In Fig. 7 it can be seen that the symmetry of the graph to its vertical axis is not preserved during the animation, whereas

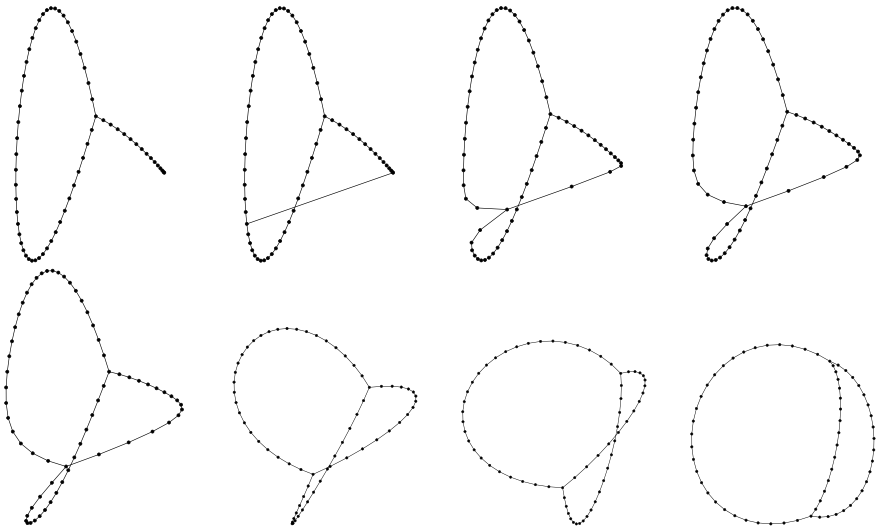


Figure 5: Update by iteration (read left to right, top to bottom). Note the spread of change along the graph structure

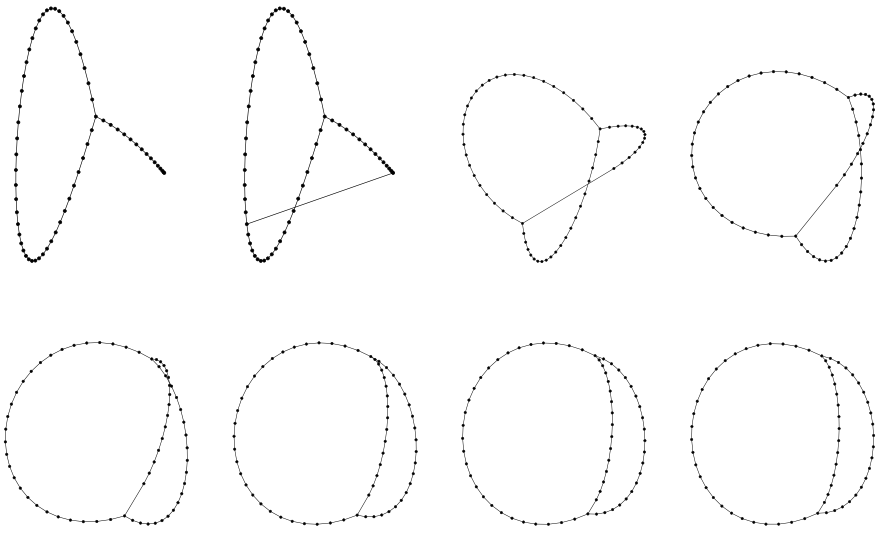


Figure 6: Update by interpolation. Layout anomalies are restricted to modified part of graph

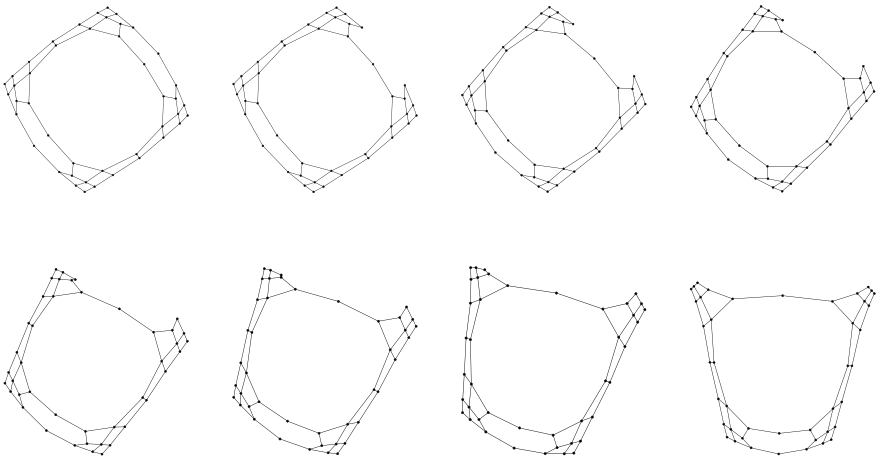


Figure 7: Update by simple linear interpolation. Intermediate layouts are less symmetric

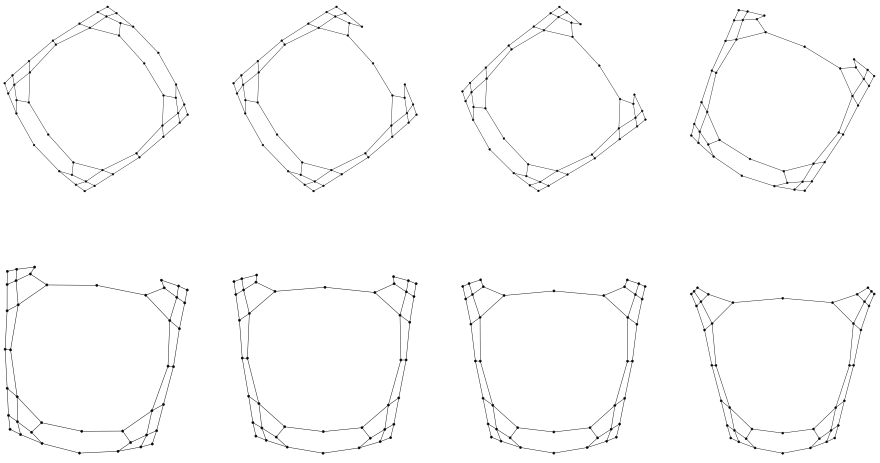


Figure 8: Interpolation updates maintain symmetry

in Fig. 8 each intermediate layout preserves this symmetry.

Figure 9 finally shows some snapshots of a small world evolving from the initial ring structure. The layouts were obtained by using matrix interpolation (one intermediate step per change shown). Note that deletion and insertion of vertices requires some extra efforts, in particular, if the deletion of a vertex disconnects the graph.

5.2 Deletion and insertion of vertices

Consider deletion of a single vertex v, that does not disconnect the graph. Matrix $L(G^{(1)})$ is then expanded by one row and column of zeros corresponding to vertex v, such that $L(G^{(0)})$ and $L(G^{(1)})$ have the same dimension. This derived matrix has a double eigenvalue 0. A new corresponding eigenvector is, e.g., $(0,\dots,0,1,0,\dots,0)^{\mathrm{T}}$, where the 1 is at position corresponding to v. This eigenvector will cause vertex v to drift away during power iteration, and thus all other vertices stick together. This can be prevented by defining $\ell_{v,v} = g$ in matrix $L(G^{(1)})$, leading to a movement of v towards 0. But in practice, the following method proved to be successful. After every matrix multiplication reset the position of v to the barycenter of its neighbors. This either prevents a drifting away or an absorbing to 0, which would otherwise be hard to manage. Apart from using matrix $(t-1)L(G^{(0)}) + tL(G^{(1)})$ for the power iteration, orthogonalization and normalization also have to be adapted. For time $t = 1$ we only need $x_1 \perp (1,\dots,1,0,1,\dots,1)$, instead of $x_1 \perp \mathbf{1}$, and only the restriction to the elements not corresponding to v have to be normalized. Both can be done by linear interpolation of these operations.

Insertion of a vertex v is treated analogously. Expand matrix $L(G^{(0)})$ by one row and column of zeros as above. Orthogonalization and normalization again have to be adapted.

5.3 Disconnected graphs

The deletion of a *cut vertex* (or a *bridge*) disconnects $G^{(1)}$ into $p \geq 2$ components $G_1,\dots,G_p$. Each component is drawn separately by spectral methods and afterwards, these layouts are merged to a layout for $G^{(1)}$. Basically, there are three parameters for each component, that have to be determined after a layout x_1 for each G_j was computed. The first one is a suitable rotation angle φ_j for each component. The second one determines the size of each component, i.e., find a constant s_j, that scales x_1 to $s_j x_1$. The third one determines the barycenter c_j of each component.

The removal of a cut vertex (or a bridge) yields a matrix $L(G^{(1)})$, that, after rearranging, consists of p blocks $L_1,\dots,L_p$, which are Laplacian matrices of lower dimensions

$$L(G^{(1)}) = \begin{pmatrix} L_1 & & & \mathbf{0} \\ & L_2 & & \\ & & \ddots & \\ \mathbf{0} & & & L_p \end{pmatrix} .$$

Each of the components is now drawn separately, simply by the common power iteration of the whole matrix $L(G^{(1)})$, where only normalization and orthogonalization have to be modified appropriately. The barycenter c_j of each component thus is 0, but will be reset later. The goal of the rotation of a component is to minimize the difference between its old and new layout, i.e., to find an angle φ_j that minimizes

$$\|x_0 - \hat{x}_1\|_2^2 + \|y_0 - \hat{y}_1\|_2^2 \,,$$

where $\hat{x}_1$, $\hat{y}_1$ denote the coordinate vectors x_1, y_1 after a rotation of the coordinates by the angle φ_j. Such problems are usually solved by a Procrustes analysis, where a singular value decomposition yields the optimal angle. But in the 2-dimensional case it can be shown with elementary means that either

$$\varphi_j = \pi - \arctan \frac{x_0 y_1 - x_1 y_0}{x_0 x_1 - y_0 y_1} \quad \text{or} \quad \varphi_j = -\arctan \frac{x_0 y_1 - x_1 y_0}{x_0 x_1 - y_0 y_1}$$

is the minimum of the optimization function. The scaling factor is set to

$$s_j := \frac{\sqrt{\eta_j}\,|G_j|}{\sum_{v \in G_j} \sqrt{x_1(v)^2 + y_1(v)^2}} \,, \quad \eta_j := \frac{|G_j|}{|G^{(1)}|} \,,$$

where $|G_j|$ denotes the number of vertices of G_j. This entails that the average node distance to the barycenter in the component j is proportional to $\sqrt{\eta_j}$. A circle around the barycenter whose radius is the scaled average distance has now area proportional to η_j. This method works well in practice since components often are round-shaped. The main idea for arranging the components in the 2-dimensional plane is to place the barycenters on a circle around the origin with radius R. A circle sector is assigned to each component with angle proportional to the number of nodes, analogous to the stretching factor. For notational purposes identify the plane with complex numbers and reset the barycenters to

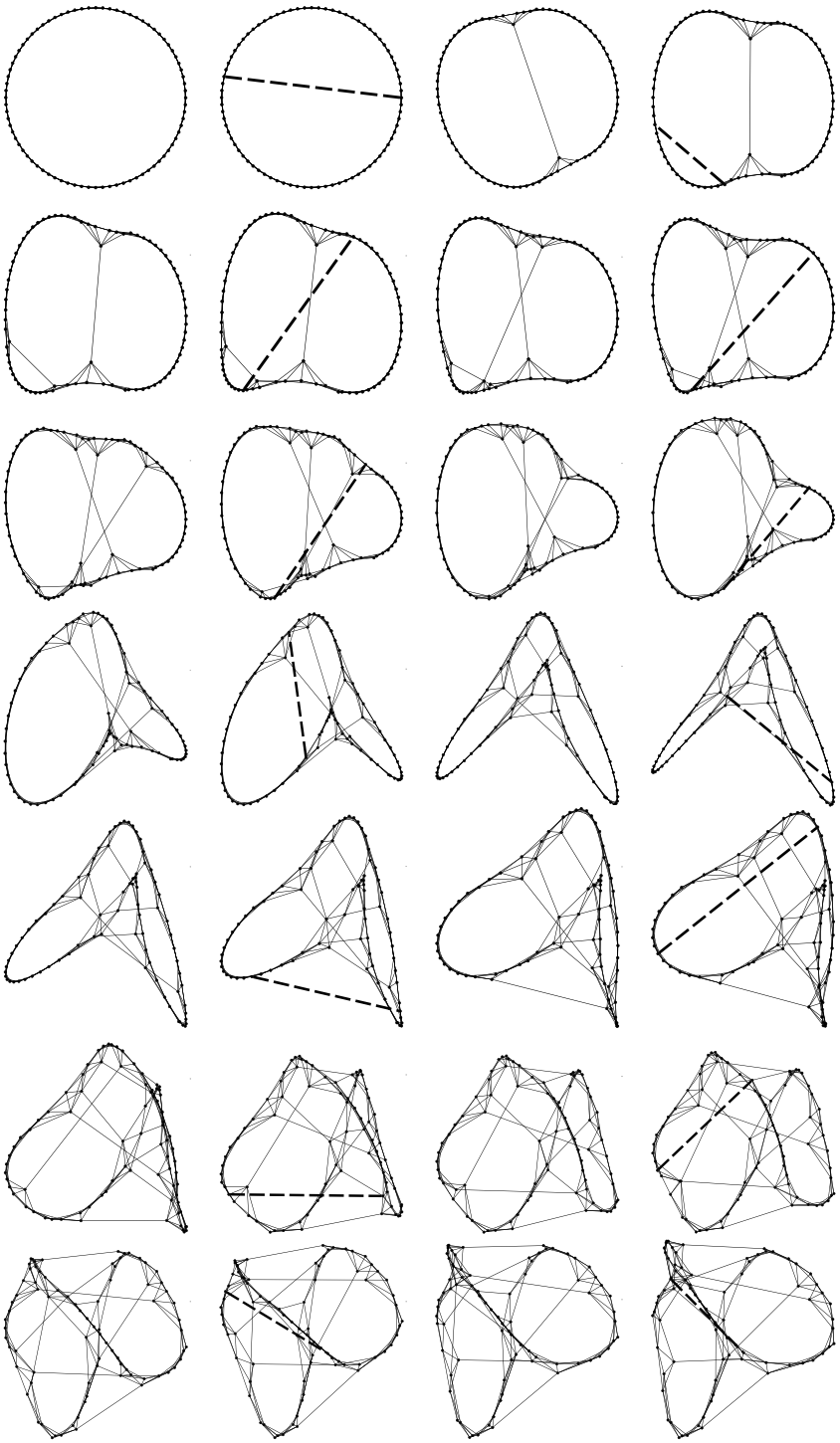
$$c_j := R \exp\Big(\frac{2\pi i}{\eta}\big(-\frac{\eta_j}{2} + \sum_{\ell=1}^{j} \eta_\ell\big)\Big) \,, \quad \eta := \sum_{j=1}^{p} \eta_j \,.$$

The radius R is chosen as

$$R := \max_{1 \le j \le p} \frac{d_j}{\sin(\pi \frac{\eta_j}{\eta})},$$

where d_j is the maximum distance from the barycenter to a node in component j. This guarantees that there is a circle around each component lying in the sector of the component, such that overlapping is prevented.

Altogether, when removing a cut vertex, power iteration with modified orthogonalization/normalization is applied for the chosen breakpoints, the components are rotated, scaled and moved linearly to their new barycenters. Further splitting and merging of connected components are handled analogously, see Fig. 10 for an example. Note that the outcome of a splitting/merging process is



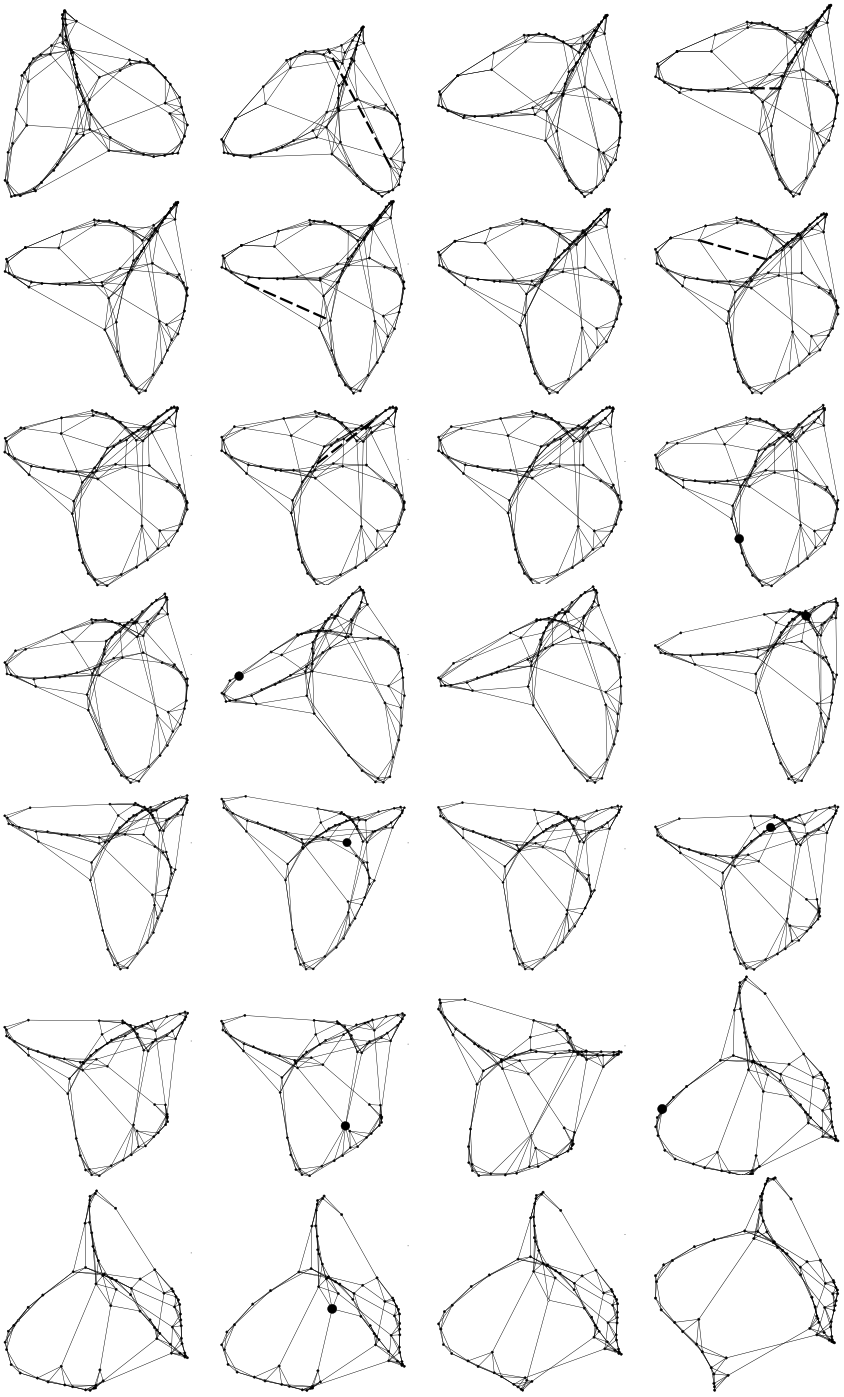


Figure 9: Evolution of a small world (read left to right, top to bottom)

assigned the same amount of area and sector as before (under the assumption that there is only a slight change in the node number). The rotation also helps recognizing the structures since unmodified components maintain their shape, size and sector.

6 Discussion

We have proposed a scheme for dynamic spectral layouts and applied it to changing small-world graphs. While there is no need to make special provisions for logical updates, it turns out that matrix interpolation is the method of choice for the physical update. Despite its simplicity, the scheme achieves both static layout quality and mental-map preservation because it utilizes stability inherent in spectral layout methods.

Much of the scheme directly applies to force-directed methods as well, and is in fact driven by common practices [8].

For both spectral and force-directed layout update computations are rather efficient since the preceding layouts are usually very good initializations for iterative methods. For large graphs, it will be interesting to generalize the approach to multilevel methods, possibly by maintaining (at least part of) the coarsening hierarchy and reusing level layouts for initialization.

In general, spectral layouts are not suitable for graphs with low connectivity, even in the static case. However, our dynamic approach is likely to work with any improved methods for static spectral layout as well.

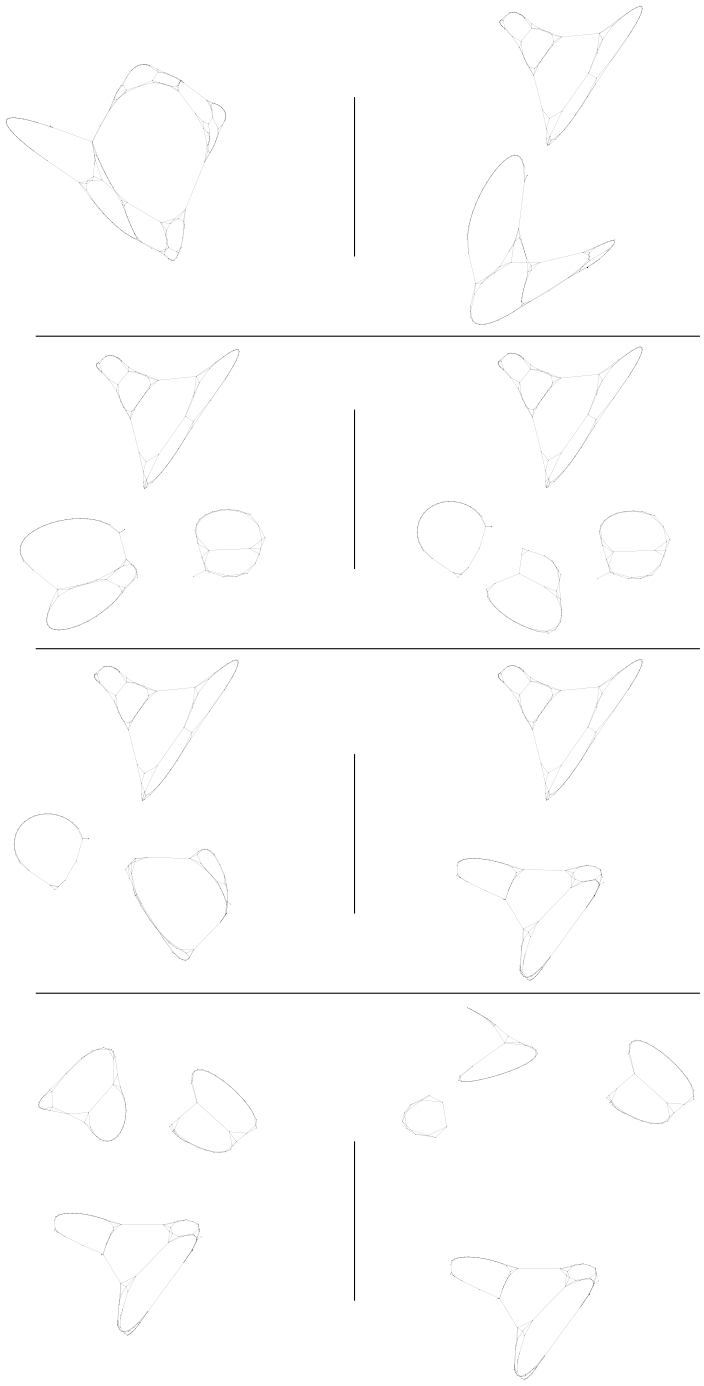


Figure 10: Drawing connected components (read left to right, top to bottom)

References

[1] U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sktech-driven orthogonal graph drawing. In *Proc. GD 2002*, LNCS 2528, pages 1–11. Springer, 2002.

[2] U. Brandes, V. Kääb, A. Löh, D. Wagner, and T. Willhalm. Dynamic WWW structures in 3D. *Journal of Graph Algorithms and Applications*, 4(3):103–114, 2000.

[3] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In *Proc. GD 1997*, LNCS 1353, pages 236–247. Springer, 1997.

[4] J. Branke. Dynamic graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs*, LNCS 2025, pages 228–246. Springer, 2001.

[5] S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. *Journal of Graph Algorithms and Applications*, 4(3):47–74, 2000.

[6] C. Demestrescu, G. Di Battista, I. Finocchi, G. Liotta, M. Patrignani, and M. Pizzonia. Infinite trees and the future. In *Proc. GD 1999*, LNCS 1731, pages 379–391. Springer, 1999.

[7] S. Diehl, C. Görg, and A. Kerren. Preserving the mental map using forsighted layout. In *Proc. VisSym 2001*. Springer, 2001.

[8] P. Eades, R. F. Cohen, and M. Huang. Online animated graph drawing for web navigation. In *Proc. GD 1997*, LNCS 1353, pages 330–335. Springer, 1997.

[9] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. GraphAEL: Graph animations with evolving layouts. In *Proc. GD 2003*, LNCS 2912, pages 98–110. Springer, 2003.

[10] C. Erten, S. G. Kobourov, and C. Pitta. Intersection-free morphing of planar graphs. In *Proc. GD 2003*, LNCS 2912, pages 320–331. Springer, 2003.

[11] C. Friedrich and P. Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6(3):353–370, 2002.

[12] C. Friedrich and M. E. Houle. Graph drawing in motion II. In *Proc. GD 2001*, LNCS 2265, pages 220–231. Springer, 2001.

[13] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

[14] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.

[15] Y. Koren. Drawing graphs by eigenvectors: Theory and practice. *Computers and Mathematics with Applications*, 49(11–12):1867–1888, 2005.

[16] Y. Koren, L. Carmel, and D. Harel. Drawing huge graphs by algebraic multigrid optimization. *Multiscale Modeling and Simulation*, 1(4):645–673, 2003.

[17] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[18] F. Rellich. *Perturbation Theory of Eigenvalue Problems*. Gordon and Breach Science Publishers, 1969.

[19] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.