A Coloring Solution to the Edge Crossing Problem

Radu Jianu[1], Adrian Rusu[2], Andrew J. Fabian[2], David H. Laidlaw[1]
[1]Department of Computer Science, Brown University, Providence, RI 02912, USA
[2]Department of Computer Science, Rowan University, Glassboro, NJ 08028, USA
jr@cs.brown.edu, rusu@rowan.edu, fabian78@students.rowan.edu, dhl@cs.brown.edu

Abstract

*We introduce the concept of coloring close and crossing edges in graph drawings with perceptually opposing colors making them individually more distinguishable and reducing edge-crossing effects. We define a "closeness" metric on edges as a combination of distance, angle and crossing. We use the inverse of this metric to compute a color embedding in the L*a*b* color space and assign "close" edges colors that are perceptually far apart. We present the following results: a distance metric on graph edges, a method of coloring graph edges, and anecdotal evidence that this technique can improve the reading of graph edges.*

Keywords--- **graphs, colors, color embeddings**.

1. Introduction

A natural visualization for graphs is the embedding of the graph into a plane, referred to as a geometric graph. A classic problem in graph drawing is how to embed a graph into the plane so that it meets certain aesthetic requirements, such as reducing edge crossings, maximizing angles between edges, minimizing the graph's area, etc, to produce readable and aesthetically pleasing geometric graphs. This problem has received considerable attention that mostly focuses on producing the actual geometric embeddings with the aesthetic constraints, while the topic and benefits of coloring the resulting geometric representations have been less well explored.

Experimental studies such as [16] have shown that edge crossings are the prevalent inhibitor in understanding graphs. However, topologies of real-life graphs usually make it impossible to yield drawings free of crossings. We therefore propose a coloring solution to the problem: edges that come in contact or that appear to be overlapping are painted with perceptually opposing colors in order to maximize the viewer's ability to tell them apart [Fig. 1]. The coloring will break involuntary Gestalt effects, make edges individually more distinguishable and disambiguate crossings.

A section on related work follows. We continue with our methods and then present our results in terms of contributions and algorithmic outputs. A discussion follows and we end with conclusions.

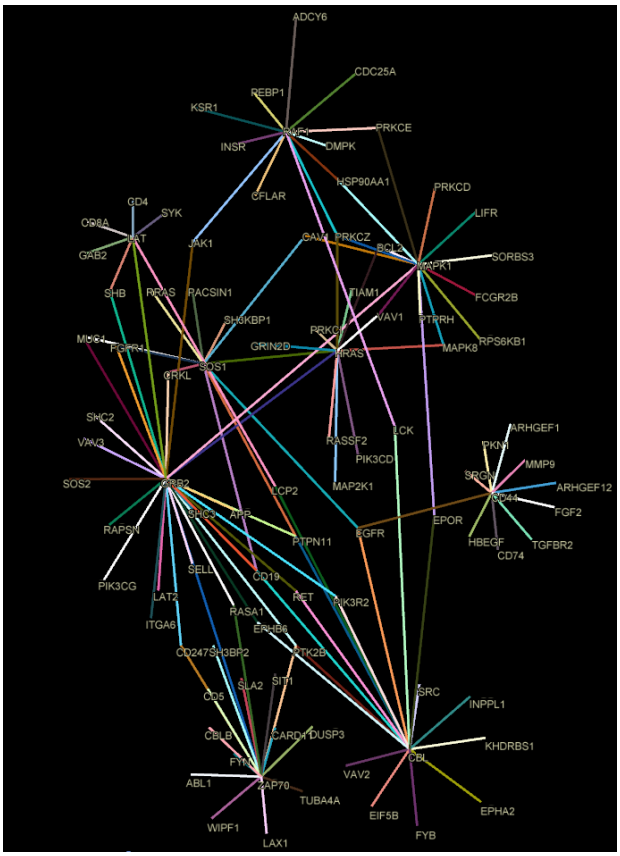

Figure 1. Protein interaction network derived from data available in the Human Protein Reference Database (HPRD) with described edge coloring solution.

2. Background

2.1. Drawing graphs

There are several algorithms available for embedding geometric graphs. Among the most popular are FD-FR by Fruchterman and Reingold, [9] based on an original idea by Eades [5]; FD-K by Kamada and Kawai, [12] also based on Eades' idea; POGB by Tamassia [21;, PG by Woods [25]; PGS by de Fraysseix et al. [8]; SEIS by Seisenberger [20]; and Tu by Tunkelang [23]. FD-FR and FD-K are force-directed algorithms which equate graph drawing to minimizing

energy in a physical system, POGB is a planar orthogonal grid algorithm, PG is a planar grid algorithm with many sloped edges, PGS uses all straight lines, SEIS uses PGS followed by compression to reduce the overall area, and Tu is an incremental algorithm whose drawing is similar to a force-directed algorithm.

In [16], some of the previous algorithms were compared for aesthetics. A user study was conducted to determine the users' performance in reading graphs, and the criteria that made the graphs easier or more difficult to read. The empirical evidence from this study suggests that edge crossings are one of the primary culprits that make graphs difficult to read.

While geometric embedding algorithms have been the primary method of addressing edge crossings, graph drawing coloring algorithms have received little attention. The traditional graph coloring problem of assigning adjacent nodes different colors does exist, but it only deals with colors in a theoretical and abstract fashion. However, there has been some research in traditional coloring of geometric graphs. Examples of some of these investigations include a study by Bern, Eppstein, and Hutchings on an algorithm for coloring quadtrees [1], colorings of geometric intersection graphs by Eppstein [6], and colorings of arrangement graphs by Felsner, Hurtado, Noy, and Streinu [7]. There have also been studies by Levkowitz and Herman [14], Robertson [18], and Ware [24] on effectively building color maps corresponding to data values in data visualization or images. Tracing a path through color space to construct a color scale has been studied by Rheingans and Tebbs [17].

From a methodology standpoint, [4] comes closest to our proposed method by introducing an algorithm for coloring neighboring nodes in a geometric graph using perceptually different colors. We are unaware of similar work for graph edges.

2.2. Color-coding similarity

Choosing appropriate colors to represent data values or using perceptually similar colors to indicate similarity between objects have received more attention. The studies in [10], [14] and [24] perform empirical studies to address the problem of generating perceptually effective color-maps. [3] uses perceptually uniform colors to underline similarities in DTI fiber tracts.

Ultimately, the problem can be reduced to producing a 2D or 3D embedding of a distance metric and immersing it into a color space. Representing a distance metric in a viewable space has mostly been researched in the context of multi-dimensional data visualization. So-called multi-dimensional scaling (MDS) techniques map points in multiple dimensions to a visualizable 2D or 3D space, while preserving the distance relations among them.

The MDS techniques fall into two categories: linear and non-linear methods. The linear methods perform linear combinations of the multiple dimensions to approximate them in lower dimensions. Two such methods are Principal Component Analysis (PCA) [11] and Star coordinates [13]. The drawbacks to these linear methods are: requiring an explicit vector representation of the points, needing to recompute the result for every change to the dataset, and a lack of interaction.

Non-linear methods solve these problems. They can usually take the distance function directly as input and define an embedding error measure to convey how well the embedding preserves the original distances. Gradient descent or force simulation can then be used to find an embedding that corresponds to a local minimum of the embedding error measure. Two such methods are Sammon's Mapping [19] and Force Directed Placement (FDP) [9], which was originally proposed by Eades [5] as an approach to graph drawing. The latter approach initially places the points at random, and then iteratively moves the points by treating them as point masses connected by springs, whose lengths are the distances that need to be embedded. The total energy of the spring forces in the graph is minimized, and this minimum energy state is the final embedding. We choose to use this approach since it accepts distances directly as input, tends to yield better embeddings than the linear methods, is iterative and thus interactive, and is easy to implement and adapt to the particularities of our problem.

An iteration of the original FDP model is $O(n^2)$, and since at least n iterations are necessary to reach equilibrium, the final complexity is $O(n^3)$. Tejada et al. [22] proposed an approach called *Force Scheme* to reduce this complexity by requiring fewer iterations to reach the final state, keeping the complexity at $O(n^2)$. Iterations with linear complexity were developed by Chalmers, [2] producing another model of overall complexity $O(n^2)$. A complexity of $O(n^{5/4})$ was achieved by Morrison et al. [15] by creating a hybrid model based on approximations using samples and interpolations.

As for color spaces, the most popular color space for display devices is RGB where the intensities of red, green, and blue are varied to produce colors. The problem with RGB is that perceived distance in color does not correlate to Euclidean distance between color points making it impractical for translating a spatial embedding into a corresponding color embedding. CIE L*a*b* (Lab) overcomes this drawback. It defines colors as unique combinations of luminosity (L), degree of magenta to green (a) and degree of yellow to blue (b). The property that makes Lab suited for color mappings is that Euclidean distances between color points in the 3D Lab space correspond to a similar perceived distance in color. Thus, preserving the inverse of the edge closeness measure in this space will yield edge colors that are perceptually far for edges that are geometrically close.

3. Methods

3.1. Edge distance

Assuming we have a geometric embedding of a graph, we define a distance measure of the set of edges. Intuitively, the distance between two edges is small if

they interact visually. We thus define the distance measure as a function of line segment distance, crossing, and angle. Specifically, we use (1), to derive values for this distance function. We then embed the inverse of the function into a color space to compute edge colors. As a result, perceptually close edges will be assigned colors that are far apart.

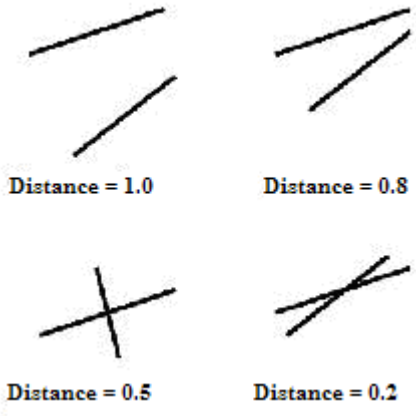

Figure 2. Examples of edge pairs and their approximate, normalized distance values.

The closeness of two edges, e1 and e2, is computed as follows:

$$D(\text{e1},\text{e2}) = \begin{cases} \theta, & dist = 0 \\ dist + 1, & otherwise \end{cases} \quad (1)$$

where *dist* is the minimum distance between e1 and e2 and $\theta = \dfrac{\text{minimum angle between e1 and e2}}{\pi/2}$

and where D(e1, e2) is the closeness metric. We found that a metric that varies linearly with edge distance and crossing angle performs better than polynomial functions of the two because it is less affected by the wide range of angles and distances common to many graph drawings.

3.2. Color embedding

Similar to [4], we compute the colors by embedding the inverse metric described in the previous section into the Lab color space using a force directed method. However, we use a different method of constraining the points to the visible Lab space.

We create a mesh approximation of the Lab gamut [Fig. 3] and use it as a 3D container for embedding. Points corresponding to edges are created and placed within the Lab gamut container, and forces are computed on them iteratively. While in traditional spring based methods, forces are translated directly into positional displacement, we chose to use a physically correct simulation and use the forces to compute velocities and use then translate these into displacements. The Lab gamut boundaries, enforced by the mesh, will act as a closed container that keeps the points inside by performing collision detection.

Points outside the container mesh are beyond the visible color spectrum and would be impossible to render. While [4] uses a combination of scaling and truncating to contain the points within the visible region of Lab, we use the physically based simulation described above. We chose this method because of two particularities of the Lab gamut: irregular shape and saturated colors close to the boundaries. Similarly to [4], we note that while it was possible to simulate a repulsive force at the container boundaries, it would need to have a steep gradient to allow points to come close to the saturated areas near the boundaries. Such a force would be hard to control in simulations. The physically based simulation allows points to bounce off the container and even slide across the container-faces to positions that minimize the system's energy. Overall, we achieve good embeddings using this simulation strategy. A graphical result of the final distance that is embedded and its corresponding embedding error is shown in Figure 4.

It is also important to note that while the objective is for close edges to be colored differently, it doesn't matter how distant edges are colored. This enables us to obtain better embeddings since the number of constraints on the system is significantly reduced, especially for large graph drawings.

For the actual embedding algorithm we use Chalmer's method [2] to reduce the complexity of one simulation iteration to $O(n^2)$. The collision detection of points with the gamut also needs to be factored in. To accelerate this process, we employ a set of successive tests based on the proximity of a point to a gamut face. We achieve interactive embeddings for moderately large graphs (1000 edges). For an example graph with 545 nodes and 880 edges, our algorithm required an average of 0.21 seconds/iteration. The number of iterations required for convergence depends on the specific graph layout.

A problem observed while testing our method was that edges were colored with the graph drawing's background color. We have thus added to the simulation algorithm the ability to add a repulsive color that is "avoided" by the embedder. This can be used to ensure that edges are not colored with the background color of the scene and remain distinguishable.

3.3 Implementation details

We have implemented our methods in C++ using the G3D extension library for OpenGL. We use QT 4.3 for user interface support.

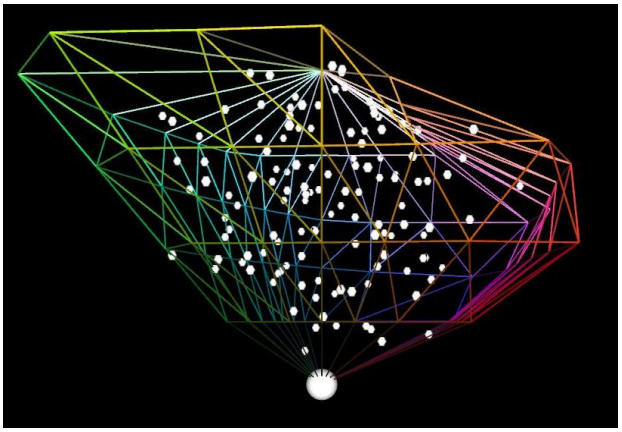

Figure 3. Mesh approximation of the Lab gamut and embedded points corresponding to edges of a graph. The white sphere at the bottom exerts a repulsive force on the points so that edge colors are picked from the lighter spectrum.

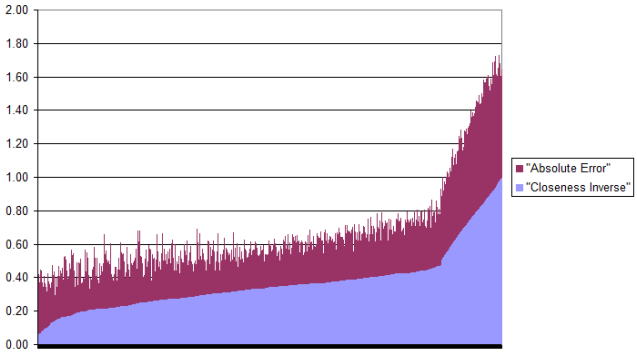

Figure 4: The inverse of the closeness metric and the absolute embedding error. The inverse of the closeness metric for each pair of edges is plotted in increasing order of magnitude with its corresponding error stacked on top.

4. Results

To the best of our knowledge the concept of assigning perceptually opposing colors to geometrically close edges in graph drawings with the purpose of making edges more distinguishable and reducing the effects of edge crossings is new. This can complement edge crossing techniques that act on the geometric embedding side and help in cases where a crossing free drawing is not possible.

We also give the first metric that defines edge closeness to define the amount of visual interaction between edges in a graph drawing. This has the potential to be used in further research and applications related to graph drawing readability.

We present a novel and efficient force-based and physically accurate algorithm that can assign perceptually opposing colors to close edges in a graph drawing. The colors cover the full spectrum and have low embedding errors.

More importantly, from a graph-reading standpoint, there seem to be several benefits to this method. For instance, lines rooted in the same node and having angles close to 0 or π are colored differently [Fig 8]. This can help the user tell them apart and understand the topology of the graph better. In the case of long edges that stem from the same node and flow in the same direction, the different colors could make it easier for users to visually follow paths. It is also likely that the method will minimize the effect of edge crossings especially in areas with high edge density.

Figures 1 and 5 - 9 show some result outputs of our algorithm. Figure 1 illustrates a real-life example: a protein interaction network extracted from the HPRD (Human Protein Reference Database) is depicted with edges colored using our technique. Despite the high visual clutter and wide range of angles and distances between edges, the algorithm manages to generate a good coloring; the upper-right region especially shows many crossing edges colored with perceptually different colors.

Figure 5 illustrates the concept in a simple example and allows us to better grasp the coloring metric. Figure 6 depicts a graph of three sparsely connected clusters with an especially good coloring. It must be noted, however, that our algorithm performs better for clustered graphs, since edges in different clusters can be co-located in the color space without any interaction. The more edges interact visually in a given area, the less distance in the Lab color space can be assigned between them, as they need to be spaced more or less evenly throughout the color space. Figure 7 illustrates this aspect: while the algorithm accommodates many of the edge crossings in a complete eight degree graph, some crossing edges are inevitably assigned similar colors.

While Figure 8 shows a relatively good coloring for a more complex graph, Figure 9 reveals another shortcoming of our technique: crossing edges on the left are deemed far apart by the algorithm compared to the many tight-angled edges on the right side of the image. They will thus be assigned a lower embedding priority. This also motivates our choice of a simple linear distance metric; a polynomial would have amplified this effect even more.

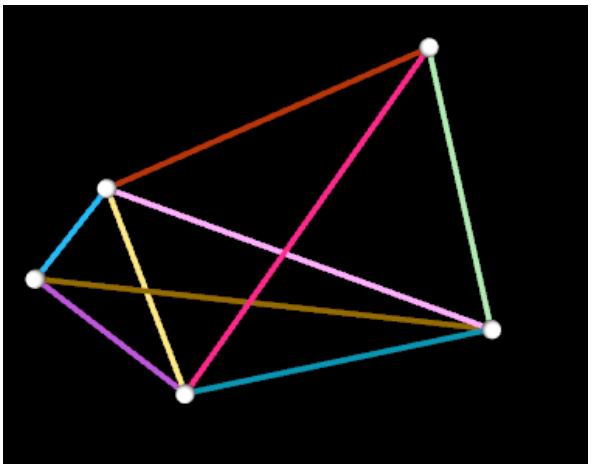

Figure 5. Simple example illustrating our coloring method. Edges crossing or stemming from the same node are assigned opposing colors.

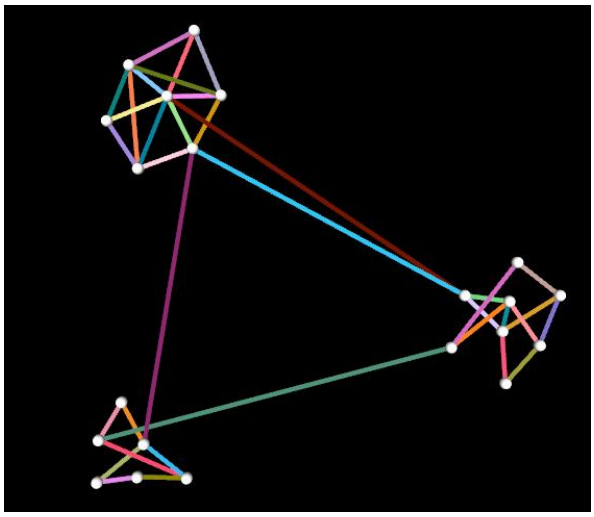

Figure 6. A graph of three sparsely connected clusters.

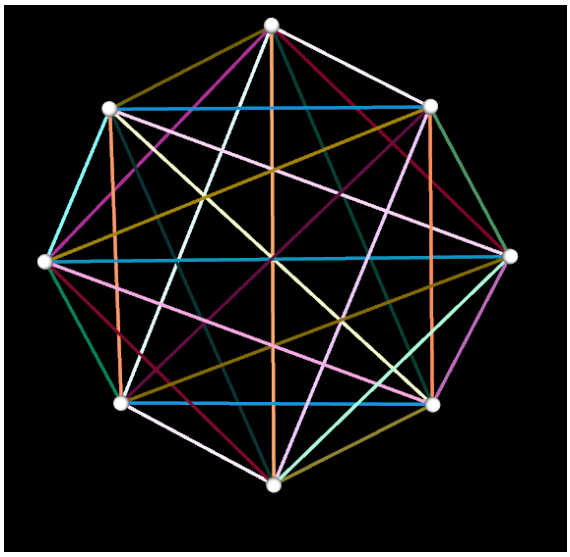

Figure 7. A complete 8-degree graph.

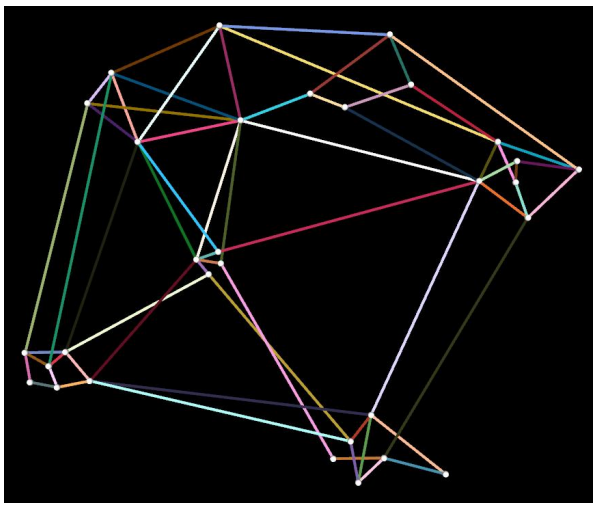

Figure 8. A more complex graph.

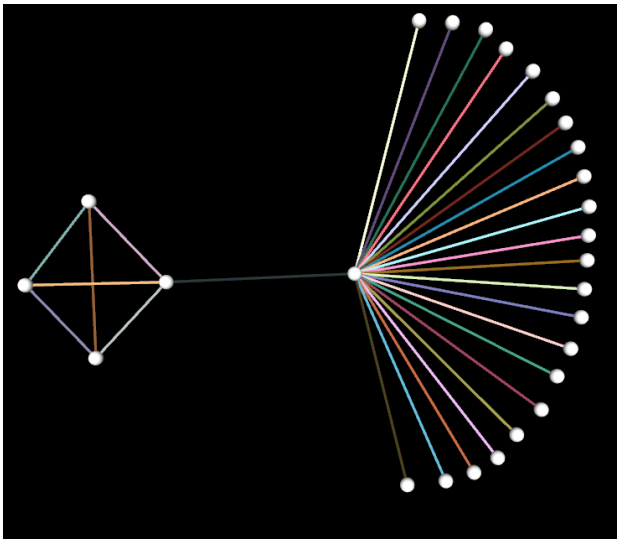

Figure 9. A problem with our coloring: the close edges in the bundle on the right cause the embedding algorithm to be insensitive to the crossing edges on the left.

5. Discussion

We show some scenarios where edge coloring helps stand out from simple examples. However, a rigorous evaluation would be helpful to quantify the improvement and determine the range of benefits. It would be particularly interesting to perform some of the studies in [16] and quantify the effects of coloring on edge crossings.

While we chose an intuitive approach to defining our "closeness" metric it would be interesting to base our choice on a more rigorous study. Unfortunately, research so far can only give us binary guidelines, in the sense that crossings do indeed affect graph-reading performance. A study on edge topologies that are hard to read in the absence of color would be interesting in itself, but would also help us refine our definition of the closeness metric.

Direct embedding in the Lab space using the gamut as a containing space yields good results. By adjusting the embedding parameters we managed to use the full visible range of Lab and obtain a low-error embedding. The alternative of using a generic embedding to obtain 3D points and rescale them to fit inside the visible Lab gamut does not produce good results because of the irregular size of the visible gamut, and the fact that the high saturation colors are located far from the gamut center.

Finally, we should note that applying color to edges in the interest of improving graph drawing readability is not always an option. Oftentimes, in real-life applications, edge color is already used to encode a value corresponding to the relationship it stands for and cannot be used for additional purposes. However, we believe there are plenty of opportunities to use our method when such constraints are not present.

Conclusions

We present a method to color close edges in graphs drawings with perceptually opposing colors. We argue this has the potential to make edges more distinguishable individually and to reduce some of the negative effects of edge crossings and edge bundles.

References

[1] M.W. Bern, D. Eppstein, and B. Hutchings. Algorithms for coloring quadtrees. *Algorithmica*, Volume 32, Issue 1, pages 87-94, 2002.

[2] M. Chalmers. A linear iteration time layout algorithm for visualizing high-dimensional data. In *Proc. of VIS'96*, pages 127-ff. IEEE CS Press, 1996.

[3] C. Demiralp, S. Zang, D. Tate, S. Correia and D. H. Laidlaw. Connectivity-aware sectional visualization of 3D DTI volumes using perceptual flat-torus coloring and edge rendering. In *Proceedings of Eurographics*, 2006.

[4] M.B. Dillencourt, D. Epstein and M. T. Goodrich. Choosing colors for geometric graphs via color space embeddings. In *Proceedings of the 14th International Symposium on Graph Drawing*, pages 294–305, 2006.

[5] P. Eades. A Heuristic for Graph Drawing. In *Congressus Numerantium,* Volume 42, pages 149-160, 1984.

[6] D. Eppstein. Testing bipartiteness of geometric intersection graphs. In *Proc. 15th Symp. Discrete Algorithms*, pages 853-861. ACM and SIAM, 2004.

[7] S. Felsner, F. Hurtado, M. Noy, and I. Streinu. Hamiltonicity and colorings of arrangement graphs. In *Proc. 11th Symp. Discrete Algorithms*, pages 155-164. ACM and SIAM, 2000.

[8] H. de Fraysseix, J. Pach, and R. Pollark. How to draw a planar graph on a grid. *Combinatorica*, Volume 10, pages 41-51, 1990.

[9] T. Fruchterman and E. Reingold. Graph Drawing by Force-Directed Placement. In *Software—Practice and Experience*, Volume 21, Issue 11, pages 1129-1164, 1991.

[10] C. Healey. Choosing effective colors for data visualization. In *Proceedings of the 7th conference on Visualization*, pages 263-ff, 1996.

[11] I. Jolliffe. Principal Component Analysis. Springer-Verlag, 1986.

[12] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, Volume 31, Issue 1, pages 7-15, 1989.

[13] E. Kandogan. Star Coordinates: A multidimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium*, pages 9-12, 2000.

[14] H. Levkovitz and G. Herman. Color scales for image data. In *IEEE Computer Graphics and Applications* Volume 12, pages 72-80, 1992.

[15] A. Morrison and M. Chalmers. A pivot-based routine for improved parent-finding in hybrid MDS. *Information Visualization*, Volume 3, Issue 2, pages 109-122, 2004.

[16] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing*, pages 248–261, 1997.

[17] P. Rheingans and B. Tebbs. A tool for dynamic explorations of color mappings. *Computer Graphics*, Volume 24, Issue 2, pages 145-146, 1990.

[18] P. Robertson. Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *IEEE Computer Graphics and Applications*, Volume 8, Issue 5, pages 50-64, 1988.

[19] J. Sammon. A Nonlinear Mapping for Data Structure Analysis. In *IEEE Trans. Computers*, Volume 13, pages 401-409, 1964.

[20] K. Seisenberger. Termgraph: Ein System zur Zeichnerischen Darstellung von Strukturierten Agenten und Petrinetzen. Diplomarbeit, Universität Passau, 1991.

[21] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, Volume 16, pages 421-444, 1987.

[22] E. Tejada, R. Minghim, and L.G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, Volume 2, Issue 4, pages 218-231, 2003.

[23] D. Tunkelang. A practical approach to drawing undirected graphs. Carnegie Mellon University, 1994.

[24] C. Ware. Color sequences for univariate maps: Theory, experiments and principles. In *IEEE Computer Graphics and Applications*, Volume 8, pages 41–49, 1988.

[25] D. Woods. *Drawing Planar Graphs*. PhD thesis, Stanford University, 1982.