Graph Multidrawing: Finding Nice Drawings Without Defining Nice*

Therese Biedl[1], Joe Marks[2], Kathy Ryall[3], and Sue Whitesides[1]

[1] School of Computer Science, McGill University
Montreal, Quebec H3A 2A7, Canada
{therese,sue}@cs.mcgill.ca
[2] MERL—A Mitsubishi Electric Research Laboratory
Cambridge, MA 02139, U.S.A.
marks@merl.com
[3] Department of Computer Science, University of Virginia
Charlottesville, VA 22903-2442, U.S.A.
ryall@cs.virginia.edu

Abstract. This paper proposes a *multidrawing* approach to graph drawing. Current graph-drawing systems typically produce only one drawing of a graph. By contrast, the multidrawing approach calls for systematically producing many drawings of the same graph, where the drawings presented to the user represent a balance between aesthetics and diversity. This addresses a fundamental problem in graph drawing, namely, how to avoid requiring the user to specify formally and precisely all the characteristics of a single "nice" drawing. We present a proof-of-concept implementation with which we produce diverse selections of symmetric-looking drawings for small graphs.

1 Introduction

Imagine you have a graph and you want a nice drawing of it. You don't know what a nice drawing for this graph looks like, but you think you can recognize one when you see it. What do you do? First, you try a known graph-drawing method. The drawing it returns is not ideal, so you modify the system's constraints or parameters or random-number seed in the hope of producing a drawing you like better. This typically results in a haphazard and tedious exploration of drawings which may or may not result in one that you like.

You might prefer instead to look at an organized selection of drawings that were chosen to show the diversity of drawings possible, subject perhaps to very general aesthetic guidelines that you supply. Then you could pick the ones that you like best, and maybe even ask the computer for more drawings similar to those. In this way, you and the computer would be collaborating in a systematic way to learn what you mean by "nice" for this graph, and to produce one or several suitable drawings. The main result of this paper is to introduce (in

* This research was supported in part by funding from NSERC and FCAR.

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 347–355, 1998.

Section 2) the *multidrawing* approach as a realization of this idealized graph-drawing system, and to present a simple proof-of-concept implementation (Section 3). Our implemented system is called SMILE (for **S**ymmetric **M**ult**I**drawing **L**ayout **E**xperiment), and it generates a diverse selection of symmetric-looking drawings for a given small input graph. Section 4 gives several directions that future experimentation and research might take, and Section 5 concludes.

2 Graph Multidrawing

Graph multidrawing is best explained operationally. The canonical multidrawing system has four principal subsystems:[1]

Layout: The layout subsystem must be capable of producing a wide variety of drawings of the same graph, either by exploiting randomization or by varying the input parameters. Fortunately, many drawing algorithms have this capability inherently (e.g., those based on spring simulation), and many other algorithms can be modified to have it.

Since many layouts of the same graph are to be generated, either the graph should be small or the layout subsystem should be fast.

Dispersion: The dispersion subsystem seeks to produce a selection of aesthetically pleasing drawings that cover the space of possible drawings of a given graph. These twin considerations of *aesthetics* and *diversity* typically require compromise: optimizing with respect to aesthetic criteria (e.g., number of edge crossings, degree of symmetry, distribution of edge lengths, etc.) usually implies a small number of optimal drawings, yet diversity implies a large number of different drawings. Thus, a good dispersion heuristic should consider aesthetics, but not optimize with regard to them.

Since an effective dispersion heuristic must also achieve diversity, a way to quantify diversity is needed. This in turn requires a way to measure the (dis)similarity of two drawings. The emergence of measuring drawing similarity as an important concept is thus one of the interesting consequences of the multidrawing approach.

Presentation: How to present as many as several hundred drawings computed by the dispersion subsystem is an important practical issue. An organized display, in which similar drawings are grouped together so that they can be browsed in a systematic fashion, is obviously preferable to an arbitrary arrangement of drawings.

Feedback: In any diverse selection of graph drawings, some will be preferred over others. The ideal feedback subsystem would give the user an easy way to request a further selection of "good" drawings, with fewer "bad" drawings included.

[1] The progenitor of the multidrawing approach is the Design Gallery™ project of Marks et al. [7], in which a similar approach is taken to a variety of computer-graphics and animation production tasks.

The next section describes our SMILE multidrawing system, which has complete layout, dispersion and presentation subsystems, but (currently) no feedback capabilities.

3 SMILE: A Proof-of-Concept Implementation

As a proof-of-concept, we have implemented the SMILE multidrawing system, which generates a diverse selection of symmetric-looking drawings for a given input graph.

We chose to experiment with the symmetry aesthetic for two reasons. First, it has been hypothesized to lead to attractive drawings [6], and yet it has been found relatively unhelpful for certain common graph-comprehension tasks [10]. Our hope was that multidrawing might contribute to the debate regarding symmetry in graph drawing. Secondly, we had available the GLIDE system [11] to use as a layout subsystem. GLIDE uses a spring-based layout algorithm that can be readily tailored to foster symmetric-looking layouts.

3.1 What SMILE Does

SMILE takes a graph as input and produces as output a large number (128 for the examples shown below) of different drawings of it. How SMILE works is detailed in the next subsection. Here we describe the interface to the presentation subsystem, and show a sampling of the 128 different drawings that the system produced for each of a few well-known graphs.

Figure 1 shows SMILE's browser interface, with which the user can inspect the computed set of drawings for a given input graph. This interface is similar to those described in detail in [2] and [7]: using a multidimensional-scaling layout method, similar drawings are located near each other in the main window; this window can be panned and zoomed interactively and interesting drawings can be viewed in their own individual pop-up windows.

Figure 2 shows several drawings of graph K_6 computed by SMILE. Although K_6 is a relatively simple graph, the drawings illustrate well the complicated interplay between aesthetics and diversity that makes graph multidrawing an interesting idea. For example, this selection shows that there is more to making pleasing drawings than just symmetry. It is clear from inspection that edge crossings, edge lengths, and vertex and edge gestalts (perceptual grouping) all play important if ill-defined and subjective roles in the way humans perceive drawings.

We have also experimented with the $K_{3,5}$ and Petersen graphs. Although only slightly larger and less uniform in structure than K_6, the variety of drawings for these graphs produced by SMILE is considerably greater, as shown in Figures 3 and 4.

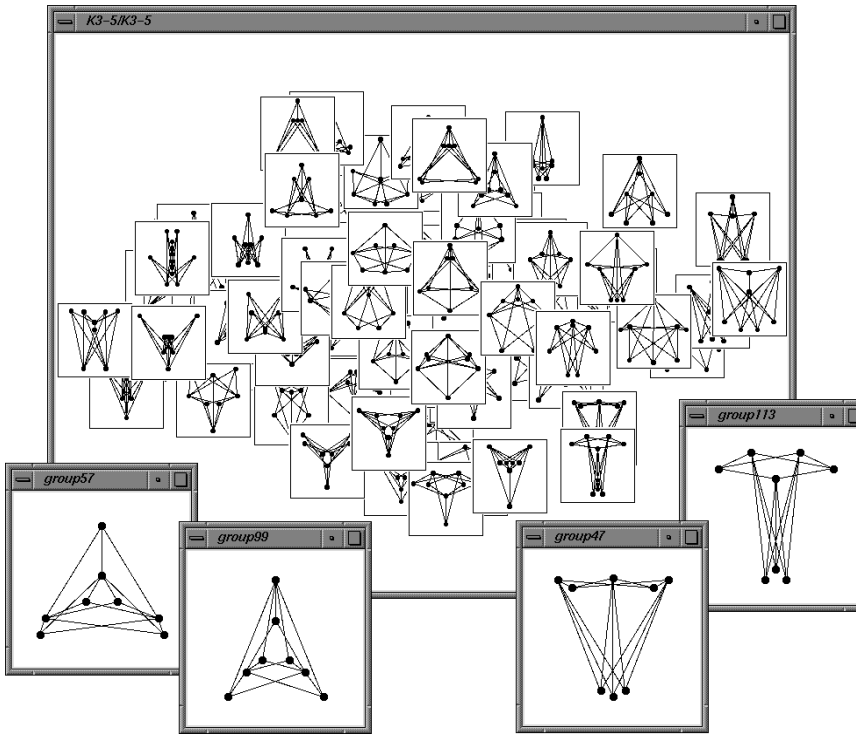

Fig. 1. The browser interface.

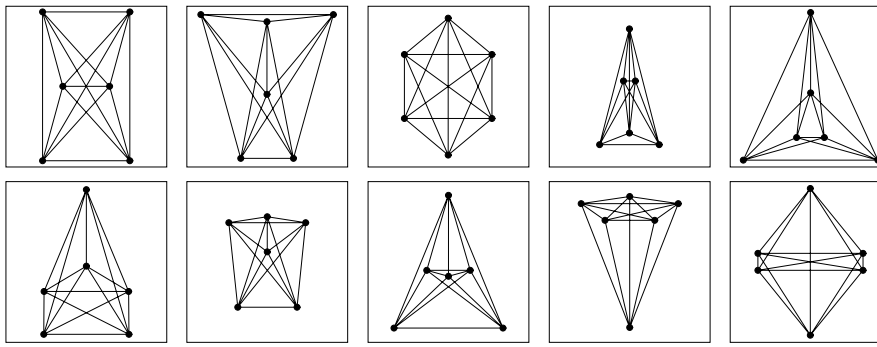

Fig. 2. Several different drawings of K_6 computed by SMILE.

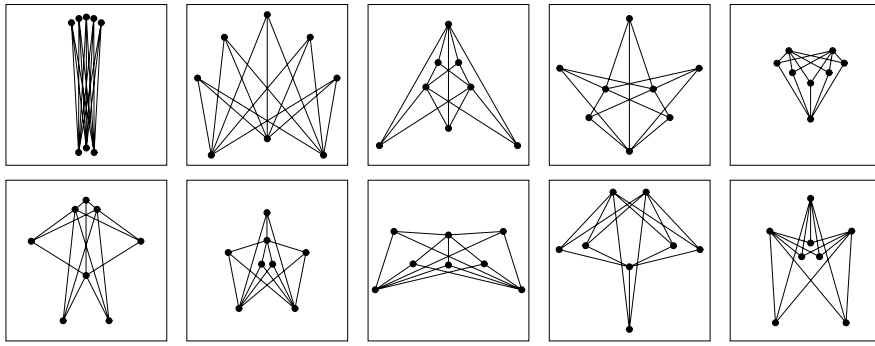

Fig. 3. Several different drawings of $K_{3,5}$ computed by SMILE.

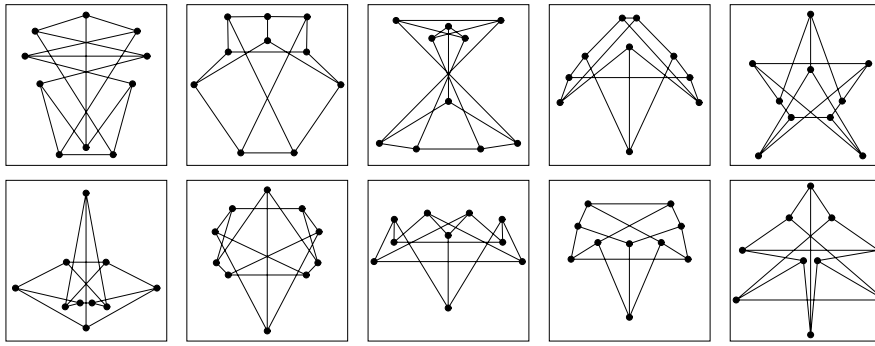

Fig. 4. Several different drawings of the Petersen graph computed by SMILE.

3.2 How SMILE Does It

Now we describe how SMILE's layout and dispersion subsystems work.[2]

Layout: The layout subsystem was derived from the GLIDE system [11]. Given an arbitrary initial configuration of a graph's vertices, it attempts to produce a straight-line drawing in which the vertices and edges are symmetric about a vertical axis, or about both vertical and horizontal axes. Given different initial configurations of vertices, it will produce different drawings. Of course it cannot achieve a perfectly symmetric-looking drawing if the initial input graph has no symmetry, and it also sometimes fails to produce a symmetric-looking drawing even when one is possible. In both such cases, the typical result is a drawing that looks fairly symmetric, with perhaps one or two asymmetric vertices or edges.

[2] The elements of the presentation subsystem are described in [2] and [7].

The underlying layout algorithm uses a generalized notion of spring force to move vertices from their initial to final positions. The spring forces are of two kinds: one kind discourages vertex-vertex and vertex-edge overlaps in an obvious fashion. The other kind encourages symmetry explicitly, unlike the implicit propensity towards symmetry that arises in more conventional spring-based systems [5]. The layout algorithm differs from the original GLIDE algorithm in two small, but significant ways. First, the computation of symmetry-inducing forces in the original algorithm requires that each vertex be matched with another vertex or with itself; in the current algorithm, vertex matchings that promote edge symmetry are favored over those that do not. Second, the spring-force simulation is run to quiescence once without vertex-edge forces in effect, and then once again with them in effect. This aids greatly in avoiding unfortunate local energy minima of the simulated physical system.

Dispersion: One way to achieve dispersion is to distill a large number of randomly generated drawings down to a small, diverse set; an alternative method is to refine repeatedly a current set of drawings so as to diversify the set. Both methods were tried by Marks et al. [7], with the latter current-set method exhibiting markedly superior performance. However, this earlier work sought only to achieve pure diversity, and did not balance diversity with aesthetics, as called for by our multidrawing application.

The dispersion heuristic used by SMILE extends the previous current-set method by considering both diversity and aesthetics (i.e., symmetry) in the set of drawings. Each member of the current set consists of both an initial vertex configuration that gets passed to the layout subsystem, and the resulting drawing that gets returned. A new candidate member can then be generated by randomly selecting an existing member of the current-set, perturbing its initial vertex configuration, and computing a new resulting drawing. The dispersion subsystem works by repeatedly generating candidate members, and substituting them for existing members whenever the substitution will improve the current set. In our case, improvement is quantified by a combination of higher symmetry scores and greater nearest-neighbor distances between drawings in the current set.[3]

Symmetry scores are straightforward: they are a weighted sum of the number of vertices and edges that are drawn symmetrically with respect to the chosen symmetry axis or axes. Given the capabilities of our layout subsystem, many drawings will have the same symmetry scores.

Devising a useful distance or similarity measure between drawings is much harder. We experimented with several measures before arriving at the following one: We greedily match compatible vertices that are at similar places in the two drawings, and sum up a measure of their distance. More precisely, for each

[3] For each graph it processed, SMILE used a current set of 128 members, and considered 5,000 candidate members. This takes from 19 minutes (for K_6) up to 95 minutes (for the Petersen graph) on a MIPS R10000 processor. See [7] for more details about the dispersion process.

vertex u in drawing D_1 we find the unmatched vertex v in D_2 such that u and v have the same set of neighbors, and such that u and v are closest according to a composite distance measure. This distance measure is a combination of the Euclidean distance between the vertices' locations in their respective drawings (normalized about the centers of their bounding boxes), and the differences in their horizontal and vertical rankings in their respective drawings. Once this matching of vertices is complete, the distance between two drawings D_1 and D_2 is just the sum of the pairwise composite distances between matched vertices.

Figure 1 illustrates anecdotally how the inverse of our distance measure corresponds to perceived similarity. The drawings in the two pop-up windows on the lower left are rated as very similar to each other, and the drawings in the pop-up windows on the lower right are also rated similar to each other. However, each drawing in the left pair is rated as having little similarity with each drawing in the right pair. Although SMILE's current similarity measure correlates positively with perceived similarity, we anticipate that deriving better measures of drawing (dis)similarity will be one of the main technical challenges of effective graph multidrawing.

4 Future Directions

In this section we speculate on possible further developments of the four components in the multidrawing architecture: layout, dispersion, presentation, and feedback.

4.1 Layout

We would like to incorporate other layout techniques into our current system architecture. Developing such techniques will typically require modification of existing algorithms to make them produce multiple different drawings of a given graph. Promising candidates include specialized algorithms, such as those for drawing trees, hierarchical graphs, and those for drawing planar graphs or subgraphs.

4.2 Dispersion

With respect to the dispersion component, the main challenge is to fine a better measure of similarity of two drawings. This is crucial not only for dispersion, but also for organizing drawings logically in the presentation subsystem, and perhaps also for exploiting user feedback (e.g., by providing useful responses to the command: "Generate more drawings like this one.").

Similarity measures may be categorized as follows:[4]

- *Topological* similarity is the similarity of the planar graphs obtained by planarizing two drawings. Two drawings are *topologically identical* if they induce the same planar graph in the same planar embedding with the same outerface ([8], p. 32).
- *Metric* similarity measures the similarity of two point sets, which for us are the drawn vertices [1]. To extend the usefulness of this technique to graphs may require matching labeled point sets, i.e., matching each point in one set with a specific point in the other set [4].
- *Positional* similarity measures the positions of nodes relative to one another, for example by computing for each pair of nodes whether they are in the same relative horizontal position in both drawings, and in the same relative vertical positions in both drawings.
- *Feature* similarity exploits the notion of prominent features, such as a few faces of big area, the shape of the convex hull, or a piece that "looks like a tail."
- *Operational* similarity can be measured by computing the number and/or magnitude of operations needed to transform one drawing to another. An operation is applied to some piece of the drawing, e.g., reflection of the piece about an edge, or a change in its proportions.[5]

4.3 Presentation and Feedback

If the only goal is to achieve diversity of a set of drawings, user feedback is of limited relevance: at most, the user might indicate a new degree or type of diversity for a future run. However, graph multidrawing combines diversity and aesthetics. The incorporation of aesthetic criteria makes it more desirable, even necessary, for the user to provide feedback. For example, once the user has selected some "nice" drawings, the system could generate a new batch with similar characteristics. To accomplish this, the system must identify which qualities the user "likes" in a drawing, and then map them onto the relevant system parameters in order to generate more drawings of the desired quality.

5 Conclusion

Instead of producing a single "optimal" drawing, a graph-drawing system should generate a diverse selection of acceptable drawings for the user's perusal. This restatement of the computer's role in the graph-drawing enterprise introduces many new challenges: modifying existing algorithms to generate multiple layouts; formalizing the notion of diversity for a set of drawings and devising heuristics to achieve it; and designing interfaces to support the user's browsing task.

[4] For another taxonomy of similarity measures, see [3].

[5] According to this categorization, the SMILE system currently uses a combination of metric, positional, and operational similarities.

Acknowledgments

Thanks to Wheeler Ruml for coding help and to François Labelle for interesting discussions about similarity measures.

References

[1] H. Alt and L. Guibas. Resemblance of geometric objects. In *Handbook for Computational Geometry*. North Holland, Amsterdam, to appear.

[2] B. Andalman, K. Ryall, W. Ruml, J. Marks, and S. Shieber. Design Gallery Browsers Based on 2D and 3D Graph Drawing (Demo). Symp. on Graph Drawing 97, *Lecture Notes in Computer Science* #1353. Springer-Verlag, pp. 322–329, 1998.

[3] S. Bridgeman and R. Tamassia. Difference Metrics for Interactive Orthogonal Graph Drawing Algorithms. In this volume.

[4] K. Imai, S. Sumino and H. Imai, Minimax geometric fitting of two corresponding sets of points. *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pp. 266–275, 1989.

[5] X. Lin. Analysis of Algorithms for Drawing Graphs. Ph.D. thesis, Dept. of Computer Science, Univ. of Queensland, Australia, 1992.

[6] R. J. Lipton, S. C. North and J. S. Sandberg. A method for drawing graphs. *Proc. 1st Annu. ACM Symp. Comp. Geom.*, pp. 153–160, 1985.

[7] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. *Proc. SIGGRAPH 97*, Los Angeles, CA, pp. 389–400, Aug. 1997.

[8] J. Manning, Geometric Symmetry in Graphs. Ph.D. thesis, Dept. of Computer Science, Purdue Univ., 1990.

[9] J. Manning, Computational complexity of geometric symmetry detection in graphs. Computing in the 90's (Kalamazoo, MI, 1989), *Lecture Notes in Computer Science*, #507. Springer-Verlag, pp. 1–7, 1991.

[10] H. Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor. *Symposium on Graph Drawing 97*, volume 1353 of *Lecture Notes in Computer Science*. Springer-Verlag, pp. 248–261, 1998.

[11] K. Ryall, J. Marks, and S. Shieber. An Interactive Constraint-Based System for Drawing Graphs. *Proc. of the 10th Annu. Symp. on User Interface Software and Technology (UIST 97)*, Banff, Alberta, pp. 97–104.