Open Problems for the Bertinoro Workshop on Graph Drawing: Visualization of Large Graphs

University of Perugia

Bertinoro, Italy, March 9-14, 2008

1 Introduction

This document describes some graph drawing problems concerned with the visualization of graphs having a big number of vertices and edges. To fix terminology, we distinguish between *large graph* and *huge graph*. Namely, we assume that a large graph fits into the main memory and then a drawing algorithm can rely on a complete knowledge of its vertices and edges. Conversely, a huge graph is so big that it cannot be stored into the main memory; in this case, a drawing algorithm can handle only a limited portion of the graph at a time. In the next sections we briefly describe some approaches for the visualization of large and huge graphs, and suggest the study of specific problems for each approach.

2 Topological Windowing

The *topological windowing* approach (also called *online graph drawing* approach) has been introduced in [5, 8] for the exploration of huge graphs. This model assumes that the user can browse a graph G by iteratively selecting a specific *focus vertex*; the drawing algorithm only knows and visualizes a small portion of the graph at a time, depending on the current focus vertex and on a limited number of previously selected focus vertices. More precisely, let k be a fixed positive integer and let Q be a queue of maximum size k. Suppose that, during the browsing of the graph, Q stores the last k focus vertices selected by the user, including the current focus vertex, which we denote by v. The portion of the graph that the algorithm knows and visualizes at each time is called a *topological window* (or *logical frame*) and mainly depends on the state of Q. It is defined as the subgraph of G induced by all vertices of Q plus all the vertices of G that have a theoretic distance at most h from some vertices of Q, for some fixed positive integer $h > 0$. We denote by $G(Q, h)$ such a topological window. For small values of k and h, the topological window will be in general a relatively small graph.

To move from a topological window to another, the user can select a new focus vertex as one of the vertices adjacent to the current focus vertex v. In general, two consecutive topological windows have in common a certain set of vertices and all the edges between these vertices, i.e., they have in common a certain subgraph G'. In order to preserve the user's mental map, the drawings of G' for the two topological windows should be similar or even identical when possible.

This motivates the following general problem: *Let $G = (V, E)$ be a graph and let $V' \subset V$. Let G' be the subgraph of G induced by V' and let D' be a drawing of G'. Compute a drawing D of G that preserves D' and that satisfies certain aesthetic requirements.* In order to initiate the study of this problem, we pose a more specific question.

Problem 1 *Let G be an outerplanar graph and let $V' \subset V$. Let G' be the subgraph of G induced by V' and let D' be a straight-line drawing of G'. Compute a planar drawing D of G with a limited number of bends per edge and such that $D' \subset D$.*

The fact that G is outerplanar is a reasonably assumption if we are browsing huge graphs that are locally very sparse (e.g., trees plus a small number of extra edges). The restriction that D' is straight-line is justified by the fact that we can imagine to add dummy vertices in place of edge bends.

As a related problem, we mention a recent paper that studies how to complete a straight-line partial drawing of a tree on a given set of points using a few number of bends per edges [6].

3 Hierarchical Approach

This is an approach used to deal with the high visual complexity of large graphs. The idea is to structure the graph into a hierarchy of *clusters*, i.e., subsets of its vertices. Then, the user can visualize the graph at different levels of detail, by expanding or collapsing each single cluster. The drawing algorithm knows the whole graph and its clusters, and it should compute a nice layout of the graph, capable to convey the clustered structure of the graph to the user. Sometimes, clusters are not given as part of the input, and the drawing algorithm is free to compute a hierarchy of clusters by itself. A graph together with a hierarchy of clusters is also called a *clustered graph*.

From a theoretical point of view, one of the most interesting problems concerned with the visualization of clustered graphs is the so called *cluster planarity* problem. Roughly speaking, a clustered graph is *cluster planar* if it can be drawn with no edge crossings and so that each cluster is represented by a simple closed region containing all and only the vertices of the cluster (and of course of its subclusters); also, no edge of the graph can cross the boundary of a cluster region twice.

For the case where any two clusters are either disjoint or properly included one in the other, testing whether a clustered graph is cluster planar can be done in polynomial time [1, 2, 7] if any cluster induces a connected subgraph; the time complexity of the testing problem is still unknown if clusters can induce disconnected subgraphs.

Since in many real world applications clusters at a same level of the hierarchy can intersect, it is reasonable to extend the definition of cluster planar graph to this case; we speak of *overlapping clustered graph* and of *overlapping cluster planarity*. The formal definition of the problem is given in [4], along with some preliminary results about testing the cluster planarity if a cluster is allowed to overlap with at most another cluster at the same level. Many basic problems are still open on this subject. We list a couple of them.

Problem 2 *Let G be an overlapping clustered graph containing three clusters that overlap at the same hierarchy level. Is there any polynomial-time algorithm to test whether G is overlapping cluster planar?*

Problem 3 *What is the complexity of testing an overlapping clustered graph for cluster planarity in each of these cases?*

- *Each cluster induces a connected subgraph;*
- *Each cluster induces a connected subgraph, and each overlap between clusters induces a connected subgraph.*

Notice that, if the clusters can induce disconnected subgraphs, we think it can be easily proved the NP-Hardness of the overlapping cluster planarity testing problem, by adopting a reduction similar to the one used for the NP-Hardness proof of the Hypergraph Planarity problem [9].

4 Filtering

Another possible approach to interact with the visualization of large graphs is to use *filtering*, i.e. the removal of some vertices and/or edges in order to reduce the visual complexity of the drawing. *Vertex filtering* can be used to reduce the size (in terms of number of vertices) of the graph to be visualized. *Edge filtering* makes the graph sparser and allows the use of some classical graph drawing algorithms (e.g. algorithm for trees, planar graphs,...).

In general terms, edge filtering works as follows for a graph $G = (V, E)$. The set E is partitioned into k sets $E_0, E_1, \ldots E_{k-1}$, in such a way that each subgraph $G_i = (V, E_i)$ $(i = 0, 1, \ldots, k-1)$ has some structural properties (e.g. planarity, acyclicity,...) or it has some meaning according to the application context; k different drawings Γ_i of the k subgraphs $G_i = (V, E_i)$ $(i = 0, 1, \ldots, k-1)$ are computed. Clearly if the k drawings Γ_i are computed independently, the user's mental map is not preserved. Thus it is natural to ask that drawings Γ_i are related in some way. A possible approach is the one based on *simultaneous embeddings*. Many variants of simultaneous embedding have been defined, but the general problem can be stated as follows: compute the drawings Γ_i in such a way that each vertex has the same position in all the k drawings and such that each drawing satisfies certain aesthetic requirements. The idea is that the user can swap from a drawing to another, while maintaining unchanged the positions of the vertices and having a nice visualization for each subgraph.

A different approach is the *multi-view approach*. The idea behind multi-view approach is to display the different drawings placing them side by side. In this case it is not required that vertices have the same position in all drawings but only that it is easy to find corresponding vertices when drawings are visualized. To better formalize this idea we introduce the concept of *matched drawings*.

Let $\mathcal{G} = \{G_i = (V, E_i) \mid i = 0, \ldots, k-1\}$, be a set of k planar graphs with the same vertex set. A matched drawing of $\mathcal{G}$ is a set of drawings $\{\Gamma_0, \Gamma_1, \ldots, \Gamma_{k-1}\}$ such that:

- Γ_i is a planar drawing of G_i;
- For each $v \in V$, points $p_0(v), p_1(v), \ldots, p_{k-1}(v)$ have the same y-coordinate, where $p_i(v)$ denotes the point representing v in Γ_i.

The following results are proved in [3]:

- There exist two isomorphic 3-connected planar graphs, both with 12 vertices, that are not matched drawable.
- There exist a 3-connected planar graph and a tree, both with 620 vertices, that are not matched drawable.
- A planar graph and an unlabeled level planar (ULP) graph are always matched drawable.
- A planar graph and a graph of the family of "carousel graphs" are always matched drawable.
- Two trees are always matched drawable.

We suggest the following open problems.

Problem 4 *The technique for pair of trees described in [3] computes drawings that can have exponential area. Is it possible to compute a matched drawing of a pair of trees with polynomial area bound?*

Problem 5 *Is it possible to compute matched drawings of a pair of outerplanar graphs?*

References

[1] P. F. Cortese, G. Di Battista, F. Frati, M. Patrignani, and M. Pizzonia. C-planarity of c-connected clustered graphs: Part II - testing and embedding algorithm. Technical Report 110, DIA, Università degli Studi Roma Tre, 2006.

[2] E. Dahlhaus. A linear time algorithm to recognize clustered graphs and its parallelization. In *LATIN'98*, volume 1380 of *Lecture Notes in Computer Science*, pages 239–248. Springer, 1998.

[3] E. Di Giacomo, W. Didimo, M. J. van Kreveld, G. Liotta, and B. Speckmann. Matched drawings of planar graphs. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *Graph Drawing*, volume 4875 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2007.

[4] W. Didimo, F. Giordano, and G. Liotta. Overlapping cluster planarity. *Journal of Graph Algorithms and Applications (special issue of APVIS 2007)*. to appear.

[5] P. Eades, R. F. Cohen, and M. L. Huang. Online animated graph drawing for web navigation. In G. Di Battista, editor, *Graph Drawing, Rome, Italy, September 18-20, 1997*, pages pp. 330–335. Springer, 1998.

[6] G. L. H. M. S. W. Emilio Di Giacomo, Walter Didimo. Point-set embedding of trees with edge constraints. In *Graph Drawing, Rome, Italy, September 18-20, 1997*, volume 4875 of *Lecture Notes Comput. Sci.*, pages pp. 113–124. Springer, 2008.

[7] Q. Feng, R. Cohen, and P. Eades. Planarity for clustered graphs. In *ESA'95*, volume 979 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 1995.

[8] M. L. Huang, P. Eades, and J. Wang. On-line animated visualization of huge graphs using a modified spring algorithm. *Journal of Visual Languages and Computing*, 9:623–645, 1998.

[9] D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987.