A New Method for Efficiently Generating Planar Graph Visibility Representations

John M. Boyer

IBM Victoria Software Lab, Victoria, BC Canada
`boyerj@ca.ibm.com, jboyer@acm.org`

1 Introduction

A planar graph ***visibility representation*** maps each vertex to a horizontal segment at a vertical position and each edge to a vertical segment at a horizontal position such that each edge segment terminates at the vertical positions of its endpoint vertices and intersects no other horizontal vertex segments. The first $O(n)$ algorithms for producing visibility representations were presented in [4, 5]. These were based on pre-processing to compute both an st-numbering and the dual of the planar graph, which were then used with the combinatorial planar embedding to produce a visibility representation. Greater efficiency is obtained in [3] by eliminating the need for the planar graph dual and by re-using the pre-computed st-numbering in the PQ-tree [1] algorithm.

Recently, the Boyer-Myrvold ***edge addition*** planarity method was presented [2]. The benefits relative to many prior methods, including simpler proof of correctness and $O(n)$ implementation, are due in part to eliminating the PQ-tree's st-numbering. Hence, a new approach was required in order to extend the efficiency and simplicity of edge addition planarity into the realm of generating visibility representations.

2 Computing Vertical Positions of Vertices

During the execution of the edge addition planarity algorithm, the vertices are assigned a relative position of 'between' or 'beyond' the depth first search (DFS) parent relative to some selected DFS ancestor. Each time a back edge is embedded, if its endpoints are in separate biconnected components of the partial planar embedding, then all components that become biconnected by the new edge are merged. Each edge along the external face that is incident to a merge point is marked so that when the edge is moved off of the external face (by embedding another back edge around it), the relative position of the merge point and one of its DFS children can be assigned. Figure 1 shows how the edge marks are made, and Figure 2 depicts how they are resolved into relative vertex placements.

The relative vertex placements assigned during planar embedding are converted in a post-processing step into a vertical ***vertex order*** using pre-order DFS tree traversal. When a vertex is visited, its ancestors have already been

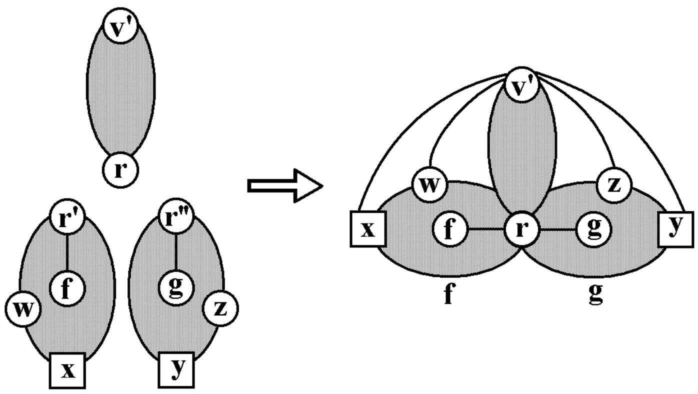

Fig. 1. When merging biconnected components, the external face edges incident to the merge points are marked with the identity of a DFS child. The children f and g are 'tied' with parent r in vertical placement until these marks are resolved.

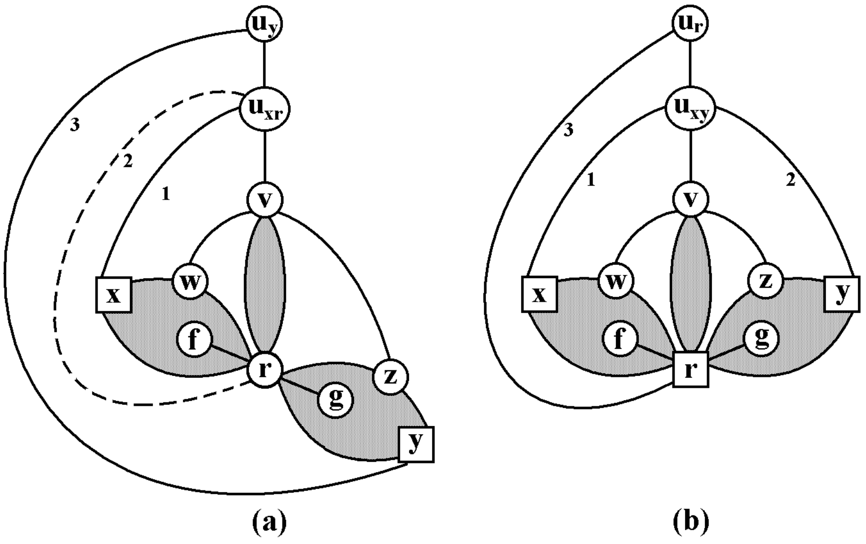

Fig. 2. (a) In a step u_x of the embedding, traversing from the descendant x of f to the parent r of f means that f is placed between the parent r and the ancestor u_x. Traversing from the parent r of g to the descendant y of g means that g is placed beyond parent r relative to ancestor u_x. (b) External activity at r can result in both children f and g being placed between r and some ancestor.

added to the vertex order. The localized information includes a marking of 'between' the DFS parent and a given ancestor or 'beyond' the DFS parent relative to the given ancestor. This is converted to be 'above' or 'below' the DFS parent, then the vertex is inserted immediately above or below its parent in the vertex order. More information is required to perform this conversion without resorting to non-linear time techniques like dynamic topological sorting. Each vertex v is positioned relative to its DFS parent p and an ancestor a, and both are added to the vertex order beforehand, but the relative positions of a and p in the vertex order are needed. Fortunately, we already store the placement of each vertex relative to its parent, and the placement of a vertex relative to its parent controls the placement of the entire DFS subtree rooted by that vertex relative to the DFS parent. Hence, the child c of a that roots the subtree containing p and v is stored during planar embedding when the relationship between v, p and a is made. Then, during this post-processing step, the relative order of p and a is obtained by query of the relative order of c and a.

3 Computing Horizontal Positions of Edges

A sweep algorithm is performed on the combinatorial planar embedding, using the vertical positions of the vertices to advance a horizontal sweep line, a data structure in which the ***edge order*** is developed. Also, each vertex keeps track of its ***generator edge*** in the edge order, which is just the first edge incident to the vertex that is added to the edge order. The generator edge provides an insertion point along the horizontal sweep line for the edges emanating from the vertex to the vertices that are below it (which have a greater vertex position number).

For starters, each edge e incident to the DFS tree root is added to the edge order according to the cyclic order in the embedding, and the generator edge of the child endpoint is set to e. For each vertex v below the DFS tree root in vertex order, we obtain the generator edge e as the starting point of the cyclic traversal of the adjacency list. The subset of edges emanating from v to vertices with greater vertex positions (i.e. below v) are added in cyclic order immediately after e. Also, for each such edge (v, w) that is added, if w has no generator edge, then (v, w) becomes the generator edge of w.

4 Conclusion

This research has yielded a new method for generating planar graph visibility representations. A linear-time reference implementation is available from the author based on the edge addition reference implementation that accompanies [2]. A pre-computed st-numbering was found to not be necessary, though the vertex ordering method produces an st-numbering as an output. It would be of theoretical interest to determine whether the notions of visibility representation and st-numbering could be completely decoupled, but first appearances suggest there would be little practical benefit as the sweep algorithm for edge ordering appears to become much more complicated with multiple source vertices. Future work would more easily find ways to compact the drawings via refinement of the algorithm presented in this paper.

References

1. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ–tree algorithms. *Journal of Computer and Systems Sciences*, 13:335–379, 1976.
2. J. Boyer and W. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.
3. R. Jayakumar, K. Thulasiraman, and M. N. S. Swamy. Planar embedding: Linear-time algorithms for vertex placement and edge ordering. *IEEE Transactions on Circuits and Systems*, 35(3):334–344, 1988.
4. P. Rosenstiehl and R. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1(4):343–353, 1986.
5. R. Tamassia and I. G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete and Computational Geometry*, 1(4):321–341, 1986.