Drawing Force-Directed Graphs Using Optigraph [*]

Jovanna Ignatowicz
Department of Computer Science
Brown University
jai@cs.brown.edu

Abstract. Optigraph is an interactive, multi-threaded tool for force-directed graph drawing. The interface allows a user to construct an arbitrary graph made of edges, free vertices and fixed vertices, and then apply circular and orthogonal spring forces to the graph to achieve an optimal graph layout. The user can step through simulations, transform the graph and vary forces at any time making the application highly interactive and educational.

1 Introduction

The primary purpose of Optigraph is to provide an implementation based on classic force-directed algorithms that encourages the user to experimentally apply various spring forces on a set of vertices and edges. The algorithms are based on work done by Eades, and Fruchterman and Reingold [1, 2]. Attracting forces are polynomial and based on the actual and ideal distances between two neighboring vertices. Repelling forces are based on the inverse square law and also consider the ideal and actual distances of two vertices, but are applied to non-neighboring vertices. An additional force is introduced to produce orthogonal drawings. It is called an orthogonal spring and redefines the ideal distance between two vertices. By this definition, two vertices are the ideal distance apart if both their x and y components are that distance apart.

2 Use of Concurrency

Optigraph achieves a high level of interactivity through the use of concurrency. It is written in Concurrent ML [4, 5], a high-level concurrent language that provides constructs to dynamically create multiple threads within one process, establish channels of communication between threads and employ various synchronization primitives. EXene [3] is the multi-threaded interface toolkit that is used on top of CML. It is unique in that the input to the interface is directed to the thread assigned to the user interface component associated with that event. Therefore,

[*] Research supported in part by the National Science Foundation under grant CCR-9423847 and by the U.S. Army Research Office under grant 34990-MA-MUR.

neither the application nor the interface is forced to wait for input to be handled by the other.

To make the best use of the concurrency at its disposal, Optigraph is organized into several parts, each of which is dominated by one or more threads. The graphics server and thread sequencer, for instance, are two infinitely looping threads. The graphics server handles requests to store, retrieve and display vertex and edge data. The thread sequencer waits for the user to start a new simulation or step through an existing one. At each step of a simulation, it spawns one thread for each vertex. These threads then calculate their new position based on their location and the location of all other vertices.

3 Results

The graphs produced by the circular spring forces greatly resemble those produced by Eades and Fruchterman and Reingold, since the algorithm is based on their work. They tend to be very symmetrical with minimal edge crossings and optimal vertex placement. The graphs produced by the orthogonal springs are orthogonal drawings that would ordinarily be produced by other methods[6].

In figure 1, an initial tree layout is shown on the left. This tree is an example of a graph constructed entirely by the user. It contains no symmetries and has varied edge lengths. The graph on the right is the result of applying circular attracting forces to the initial tree while specifying some natural distance for the edges. Although the tree is not much more symmetric, it does have uniform edge lengths. The result of adding repelling circular forces to the right graph in figure 1 is shown in figure 2. Here, the symmetries of the tree are apparent while the edge lengths stay the same. Finally, figure 3 is the result of applying orthogonal attracting and repelling forces to figure 2 to achieve an orthogonal drawing of the tree. It is interesting to note that the edge lengths of the graphs in figures 2 and 3 are longer than those in the right graph of figure 1 although the same natural distance is specified. This is due to the fact that the repelling forces of the vertices are stronger than the attracting forces of the edges. Optigraph provides a slider so that the user can modify the ratio of these two forces while they are both being applied.

Acknowledgements

I wish to thank Roberto Tamassia for his guidance and enthusiasm, Emden Gansner for introducing me to SML, CML and eXene, and Jeffrey Korn for his suggestions and technical support.

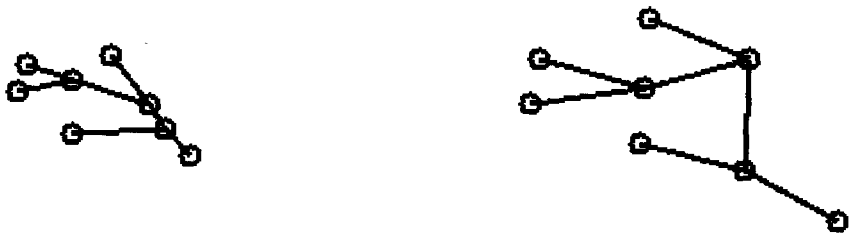

Fig. 1. Left- initial graph. Right- attracting circular forces

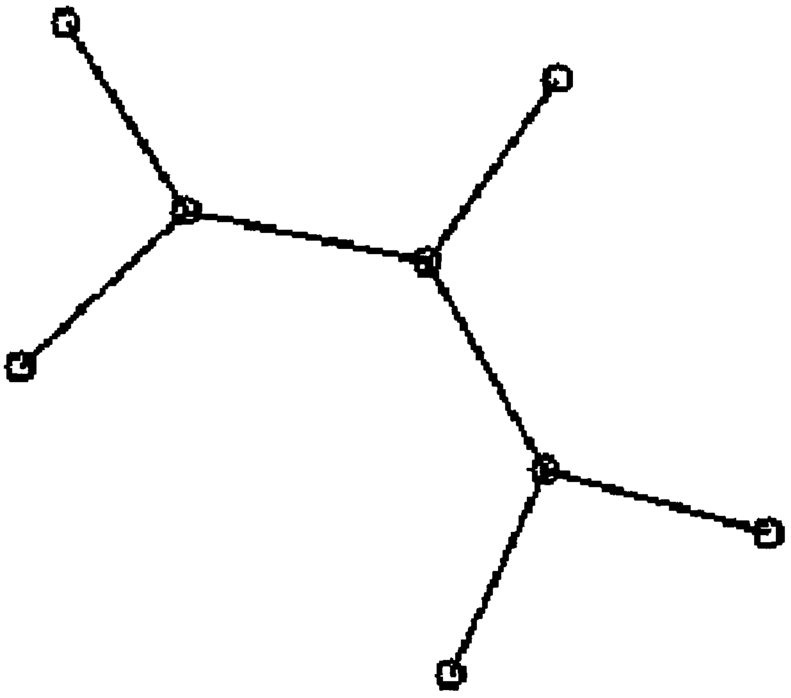

Fig. 2. attracting and repelling circular forces

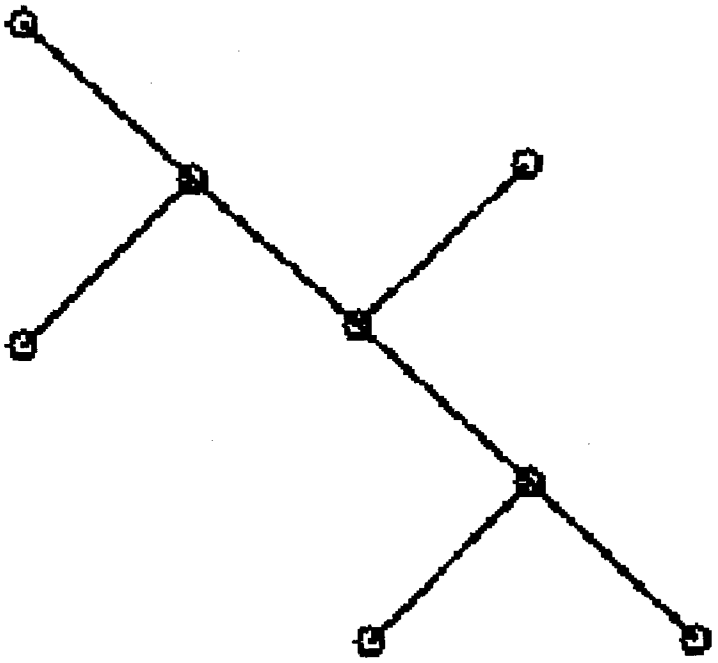

Fig. 3. attracting and repelling orthogonal forces

References

1. Eades, P., "A Heuristic For Graph Drawing", *Congressus Numerantium*, 42, pp. 149-160, 1984.
2. Fruchterman, T., Reingold, E., *Graph Drawing by Force Directed Placement*, UIUCDCS-R-90-1609, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1990.
3. Gansner, E. and Reppy, J., *eXene*, CMU Workshop on SML, 1991.
4. Reppy, J., *Concurrent Programming with Events - The Concurrent ML Manual*, Department of Computer Science, Cornell University, Ithaca, NY, 1990.
5. Reppy, J., "CML: A Higher-order Concurrent Language", *Proceedings of the SIGPLAN '91 Conference on Programming Language Design and Implementation*, June 1991, pp. 293-305.
6. Sugiyama, K. and Misue, K., "A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm", *Proceedings of Graph Drawing '94*, 1994, pp. 365-376.