3D Graph Drawing with Simulated Annealing*

Isabel F. Cruz[1] and Joseph P. Twarog[2]

Department of Electrical Engineering and Computer Science
Tufts University
Medford, MA 02155, USA

Abstract. A recent trend in graph drawing is directed to the visualization of graphs in 3D [1, 5, 6]. A promising research direction concerns the extension of proven 2D techniques to 3D. We present a system extending the simulated annealing algorithm of Davidson and Harel [2] for straight-line two-dimensional drawings of general undirected graphs to three dimensions. This system features an advanced 3D user interface that assists the user in choosing and modifying the cost function and the optimization components on-line.

1 Introduction

The 2D simulated annealing algorithm of [2] starts with an initial configuration obtained by assigning random x and y coordinates to each vertex of the graph. The algorithm proceeds iteratively by computing the cost $c(\sigma)$ of the current configuration σ and randomly selecting a new configuration σ' from the *neighborhood* of σ. The new configuration is chosen if $c(\sigma') \leq c(\sigma)$ or if a random value between 0 and 1 is smaller than $e^{(c(\sigma)-c(\sigma'))/T}$ where T is the current temperature. This iterative process continues until a termination condition is satisfied. The higher T is, the greater the probability that an "uphill" move (i.e., a move that will make the temporary solution worse) is taken. As time goes by T decreases and the configuration stabilizes.

In [2], the *neighborhood* of a configuration contains all configurations that differ from that configuration by one vertex only, and the cost function is the weighted sum of five terms. Each term is a function of an optimization component. The five different components relate to *vertex distribution*, *borderline proximity*, *edge length*, *number of edge crossings*, and *distance between vertices and edges*. The cost function penalizes non-uniform distributions of vertices, vertices that are close to the borderline of the drawing area, long edges, large number of edge crossings, and small distances between vertices and edges. The normalizing factors and the term functions are chosen so that the cost function decreases as the perceived quality of the drawing improves.

2 3D Simulated Annealing

When designing a three-dimensional simulated annealing system, some extensions are straightforward, e.g., perturbing a point within a sphere instead of a

* Email addresses of the authors: {isabel,jtwarog}@cs.tufts.edu.

circle. The choice of components for the cost function is however not as immediate. First, what constitutes a good 3D drawing of a graph is not well understood. Secondly, the possibility of changing the viewpoint that is facilitated by the interface will diminish the relevance of edge crossing (this is reinforced by the fact that we are not considering a grid as in [1], thus decreasing the likelihood of two edges crossing).

While distances between vertices and edges are still relevant, we believe the corresponding component (together with the component that relates to the number of edge crossings) is subsumed by a new optimization component that we propose, which measures the distances between pairs of (non-adjacent) edges. We call this component *edge distribution*. Further experimentation is needed to substantiate this choice as in [3, 4] for 2D graph drawing. So that this study can be made, it is paramount to determine what are the most suitable aesthetic criteria in 3D graph drawing. Because the drawings start from a random configuration and are therefore not influenced by preconceived aesthetic criteria, effective but unconventional drawings may be obtained that will contribute to the understanding of the relevant components in three-dimensional drawing.

3 The Interface

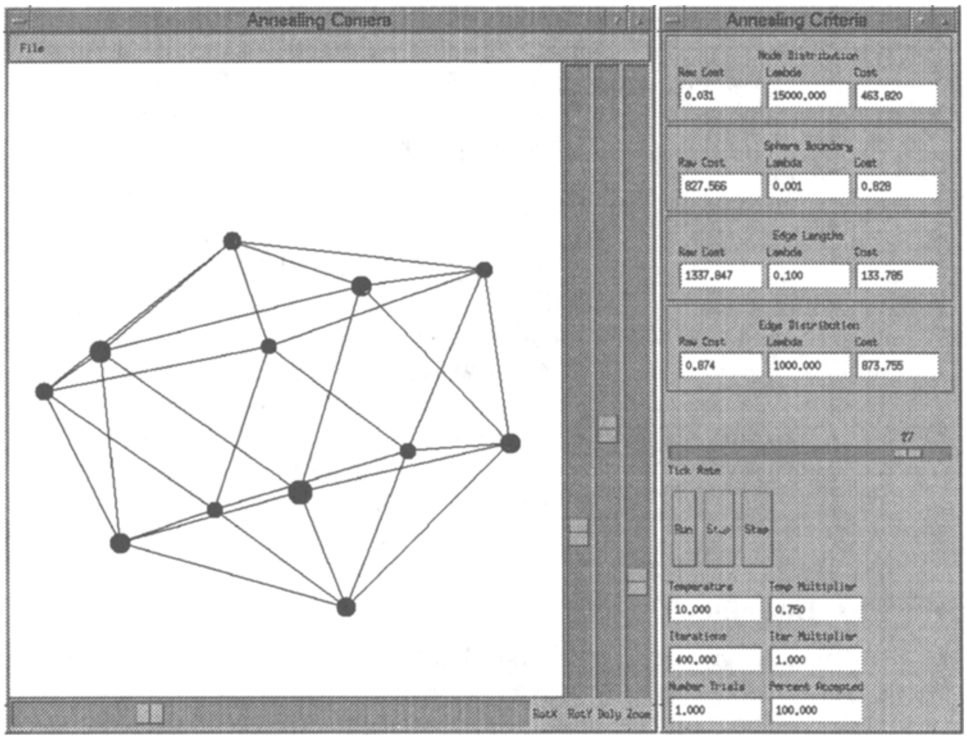
In visualizing 3D graphs, an important interaction aspect resides on the ability to dynamically change the view of the graph. The interface to our system (depicted in Figure 1) provides two primary windows. The UVN synthetic camera display allows one to change the orientation of the graph throughout the progressive development of the drawing, lending insight to the graph structure. The other window allows parameters of the annealing process as well as the cost function normalizing factors to be adjusted dynamically. The interface allows for the specification of on-the-fly constraints via the selection window, which incorporates basic spreadsheet functionality (see Figure 2). Specifically, the user can modify the temperature of each individual vertex: by appropriately lowering the temperature of a vertex, its movement will be constrained. Conversely, by raising the temperature those vertices perceived to be inadequately positioned may be "re-annealed".

The application has been implemented according to the object-oriented paradigm with C++ and Motif; it is built upon the framework established by Young [7]. As a result, it is quite extensible and is able to serve as a proper test-bed. For example, adding a new criterion requires only the instantiation of a new subclass consisting of a single cost member function. Parameter controls for such a new cost will be automatically added to the interface.

Future work includes experimentation with alternative cost functions on an extensive suite of graphs.

References

1. R. F. Cohen, P. Eades, T. Lin, and F. Ruskey. Three-dimensional graph drawing. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume



Node Distribution

Raw Cost	Lambda	Cost
0.031	15000.000	463.820

Sphere Boundary

Raw Cost	Lambda	Cost
827.566	0.001	0.828

Edge Lengths

Raw Cost	Lambda	Cost
1337.847	0.100	133.785

Edge Distribution

Raw Cost	Lambda	Cost
0.874	1000.000	873.755

Tick Rate 27

Temperature	Temp Multiplier
10.000	0.750

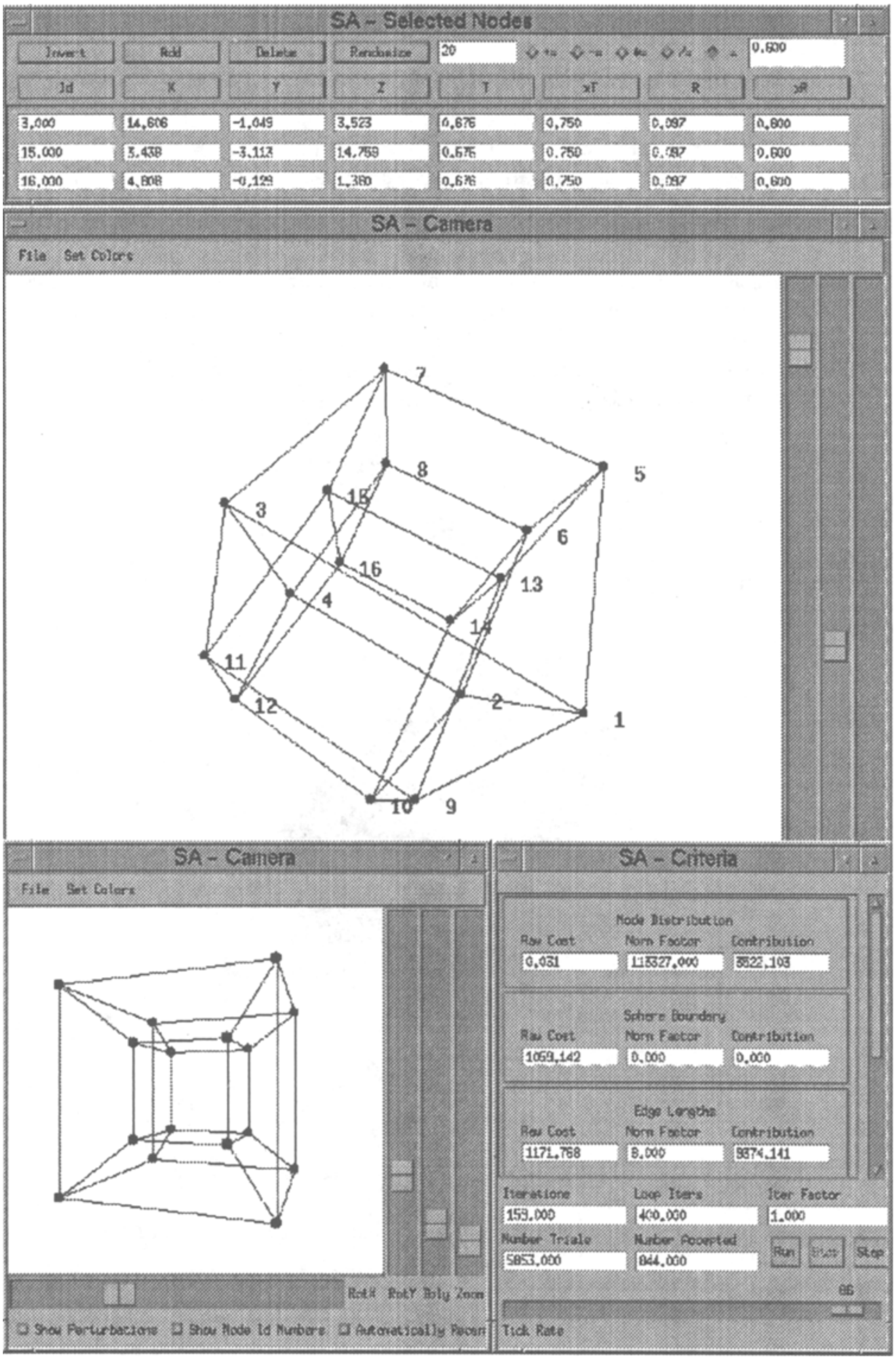
Iterations	Iter Multiplier
400.000	1.000

Number Trials	Percent Accepted
1.000	100.000

Figure 1: The graphic user interface.

894 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1995.

2. R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *Commun. ACM.* To appear.
3. G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of three graph drawing algorithms. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 306–315, 1995.
4. M. Himsolt. Comparing and evaluating layout algorithms within GraphEd. *J. Visual Languages and Computing* (special issue on Graph Visualization, edited by I. F. Cruz and P. Eades), 6(3), 1995.
5. T. Jéron and C. Jard. 3D layout of reachability graphs of communicating processes. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 25–32. Springer-Verlag, 1995.
6. S. P. Reiss. An engine for the 3D visualization of program information. *J. Visual Languages and Computing* (special issue on Graph Visualization, edited by I. F. Cruz and P. Eades), 6(3), 1995.
7. D. A. Young. *Object-Oriented Programming with C++ and OSF/Motif.* Prentice Hall, Englewood Cliffs, N.J., 1992.



20

0,600

Id	X	Y	Z	T	xT	R	xR
3,000	14,606	-1,049	3,523	0,676	0,750	0,097	0,600
15,000	3,438	-3,113	14,759	0,675	0,750	0,097	0,600
16,000	4,808	-0,129	1,380	0,676	0,750	0,097	0,600

Node Distribution

Raw Cost	Norm Factor	Contribution
0,031	113327,000	3522,103

Sphere Boundary

Raw Cost	Norm Factor	Contribution
1059,142	0,000	0,000

Edge Lengths

Raw Cost	Norm Factor	Contribution
1171,768	8,000	9374,141

Iterations	Loop Iters	Iter Factor
153,000	400,000	1,000

Number Trials	Number Accepted
5853,000	844,000

Tick Rate 86

Figure 2: A running simulation.