Graph Layout Using a Genetic Algorithm

André M.S. Barreto and Helio J.C. Barbosa
Laboratório Nacional de Computação Científica
C.P. 95113 CEP: 25651-070
Petrópolis - RJ, Brasil
e-mail: hcbm@lncc.br

Abstract

In this work we report on our experiences with applying genetic algorithms to the graph drawing problem. The automatic generation of drawings of graphs has important applications in key computer technologies. We are interested here in producing aesthetically-pleasing two-dimensional pictures of undirected graphs drawn with straight edges. To do so we use a hybrid process, applying concepts inspired by the force-directed placement technique with several aesthetic criteria which are to be minimized together. The optimization algorithm used is the genetic algorithm.

1. Introduction

Graphs are data structures widely used in many information systems, such as software engineering, database design, visual interfaces and neural networks. A graph $G = (V, E)$ is formed by a set V of vertices (with cardinality $|V|$) and a set of edges E (with cardinality $|E|$) in which an edge connects a pair of vertices. It may be represented in different styles according to the purposes of the presentation. We are interested here in producing aesthetically-pleasing two-dimensional (2D) pictures of undirected graphs drawn with straight edges. Vertices will be drawn as points in the plane inside a rectangular frame and the problem thus reduces to finding the coordinates of such points, since the edges are defined by a given 0/1 connection matrix.

Many aesthetic criteria can be conceived of and some generally accepted ones include: (i) uniform spatial distribution of the vertices, (ii) minimum number of edge crossings, (iii) uniform edge length, (iv) to exhibit any existing symmetric feature and (v) vertices should not be placed too close to edges. Of course one would strive for a fast and robust algorithm which attempts to fullfil all the aesthetic criteria simultaneously. It is clear however that one faces a multi-objective optimization problem with incomensurable and often conflicting objectives (aesthetic criteria) and, to the best of the authors knowledge, all techniques in the literature which use an optimization procedure try to aggregate the distinct aesthetic criteria into a single objective function thus avoiding a true multi-objective formulation.

The remainder of the paper is organized as follows. In Section 2 a brief literature review is presented, followed by the description of the proposed algorithm in Section 3. Section 4 summarizes some numerical experiments and Section 5 gives some conclusions and outlines the research in progress.

2. Previous work

Automatic graph layout is a long-studied problem in computer science with a large literature (see [1], for a bibliography) but we will focus on some of those references which are more closely related with the work presented here.

Inspired by the force-directed placement technique for VLSI[14], Eades[7] introduced the idea of using a mechanical model with edges replaced by springs and vertices replaced by steel rings in order to define a potential energy function which is to be minimized. However, he did not follow Hookes's law (which states that the force in the spring is proportional to the deviation from its original length) and considered repulsive forces between any pair of vertices together with attraction forces acting only between connected vertices.

Kamada and Kawai[10] established a more mechanically realistic model with the following potential energy function

$$\mathcal{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (|p_i - p_j| - l_{ij})^2 \qquad (1)$$

where $n = |V|$, p_i is the position vector of vertex i, k_{ij} and l_{ij} are respectively the spring force and the ideal distance between vertices i and j. It is clear that the idea is to get the Euclidean distance between two vertices as closer as possible to the ideal "graph theoretic" distance, which is defined

1522-4899/00 $10.00 © 2000 IEEE

by them as

$$l_{ij} = Ld_{ij} \tag{2}$$

where d_{ij} is the distance between vertices i and j measured as the length of the shortest path between them and L is the ideal length of an edge in the graph, given by

$$L = \frac{L_0}{\max_{i<j} d_{ij}} \tag{3}$$

where L_0 is the length of a size of the square display area. Finally, the spring constant in Eq. (1) is given by

$$k_{ij} = \frac{K}{d_{ij}^2} \tag{4}$$

where K is a constant. The energy function is minimized by means of the Newton-Raphson method where only a single vertex is allowed to move at a time. These concepts were explained in more detail because they were used in our algorithm, as described later.

The Fruchterman and Reingold[9] heuristic is based in [7] and also inspired in particule physics. The procedure has as guidelines: (i) connected vertices should be drawn near each other and (ii) vertices should not be drawn too close to each other. Differently from the other techniques, all vertices are allowed to move during a given iteration.

Fruchterman and Reingold also considered developing the layout in three-dimensional (3D) space and then projecting the result onto the 2D plane. They created a procedure to do so with options to choose the view reference point, view normal vector, view up vector and parallel or perspective projection. They remark that a better interface would allow the user to interactively manipulate the object in the 3D space. Finally they observed that if a graph did not appear to be 3D when laid out by the 2D algorithm, in general it would not benefit from being laid out in 3D space and then projected onto the 2D plane. Kumart and Fowler also proposed a 3D version of the spring algorithm. For more details, see [12].

Davidson and Harel[5] presented a new approach to the graph layout problem. They created a function which includes terms for vertex distribution, edge-lengths, edge-crossings and vertex nearness to borders, with different weights in each term, allowing for varying emphasis on each aesthetic criterion. For the optimization of the resulting function, the simulated annealing technique[11] was adopted. Furthermore, a fine-tuning post-processing procedure was introduced.

Coleman and Parker[4] algorithm combines the flexibility of the criteria optimization approach of Davidson and Harel with the relative speed of the method of Fruchterman and Reingold. In addition, they created several new aesthetic criteria to support new layout styles.

Among the works applying GAs to the problem of graph drawing are [8] and [3]. Eloranta and Mäkinen[8] present a GA for drawing graphs with vertices over a grid and use several operators but remark on the lack of a good crossover operator. Also they only use the number of crossing edges as an aesthetic criterion for the (few) examples presented. Branke et. al.[3] use the spring model applied to each new offspring in order to improve it. Several criteria were tried and a parallel implementation was used in order to reduce the processing time.

3. The algorithm

The proposed algorithm uses several aesthetic criteria which are to be minimized together with the energy function given by Eq. 1. This is due to the observation that the chosen energy function, which often leads to uniform edge length and uniform vertex distribution, is not always able to achieve a good solution. The objective function is thus composed by a weighted sum of the adopted criteria and the energy function $\mathcal{E}$. The weights are defined by the user and it is important to note that they can be adjusted "on the fly", that is, while the algorithm is running. In this way, the user can, for instance, reduce the number of edge crossings (by using a large initial value for the corresponding weight) and then modify the weights in order to improve other aesthetic criteria.

The criteria adopted are listed in Table 1, with the abbreviation used in the remainder of the paper, a short description and the complexity associate with their computation. Observe that each of these is to be minimized.

The optimization algorithm used here is a generational genetic algorithm (GA) where each candidate solution is encoded as a real vector containing the coordinates (x_i, y_i) of each vertex of the graph. The initial population is randomly generated and the selection process is rank-based. Two crossover operators were implemented (both of them applied sequentially with a probability of 0.85):

The **absolute crossover – AX** operator generates two new graph layouts by randomly selecting one vertex and exchanging the corresponding coordinates between the parent graphs.

The **relative crossover – RX** operator generates two new graph layouts by randomly selecting two vertices v_i and v_j in the parents graphs and calculating the corresponding vertices in the offspring by

$$v_i^{d1} = v_i^{d1} + \frac{(v_j^{p1} - v_i^{p1}) + (v_j^{p2} - v_i^{p2})}{2} \tag{5}$$

$$v_j^{d2} = v_j^{d2} + \frac{(v_i^{p1} - v_j^{p1}) + (v_i^{p2} - v_j^{p2})}{2} \tag{6}$$

Criterion	Abb.	Description	Complexity
Energy	En	Potential energy $\mathcal{E}$	$\Theta(\|V\|^2)$ [a]
Edge Crossings	Ec	Number of intersections between edges	$\Theta(\|E\|^2)$
Edge Deviation	Ed	Difference between edge lengths	$\Theta(\|E\|^2)$
Edge attraction	Ea	Distance between connected vertices	$\Theta(\|E\|)$
Vertices Repulsion	Vr	Inverse of distance between vertices	$\Theta(\|V\|^2)$
Centripetal Attraction	Ca	Distance between vertices and the centroid of layout	$\Theta(\|V\|)$
Central Uniformity	Cu	Difference between vertices distances to the centroid of layout	$\Theta(\|V\|^2)$
Vertice-edge Distance	VEd	Inverse of distance between vertices and edges	$\Theta(\|V\|\|E\|)$

[a] Assuming that the parameters d_{ij} are previously calculated and stored.

Table 1. Aesthetics crtiteria adopted in the algorithm

where v_i^{d1} refers to the first descendant's v_i vertex and v_i^{p1} is vertex v_i of the first parent.

The mutation operators are applied sequentially (and independently from crossover) each with a probability $p_m \simeq 0.25$. They are defined as follows:

The **single vertex mutation – SV** perturbs the coordinates of a randomly chosen vertex by an amount not larger than L.

The **exchange vertex mutation – EV** exchanges two randomly pre-selected vertices of the same graph.

Other operators were tested, but these ones showed to be more effective. It is particularly difficult to develop crossover operators which take into account the relation between vertices (like **RX**), and not just their absolute positions (**AX**). Another difficulty to be considered is known as the *competing conventions problem* in the area of evolutionary neural networks. In our case, it is characterized when two (or more) identical layouts of the same graph are either shifted, rotated or inverted. It would be desirable for a crossover operator that when two equivalent layouts were combined, the offspring generated would be similar to its parents. We intend to develop such an operator, using ideas from [3].

4. Computational Experiments

This section contains a series of examples that demonstrate the different aspects of the algorithm. All of them were generated with a population size of 100 individuals. The best individual at each generation was copied to the next one (*elitism*).

We start with an example that shows our algorithm in action. Figure 1(a) is a typical input graph. Figures 1(b) and 1(c) illustrate the process after 100 and 200 generations, respectively. Finally, Figure1(d) is the output produced by the algorithm. The weights of the aesthetics criteria used were $En = 0.2$, $Ec = 0.15$ and $VEd = 0.2$.

Our next examples show several drawings of the same graph, to illustrate the effect of using different aesthetics criteria. Each figure is labeled with the weight of each criterion used to draw it and the reference where it first appeared.

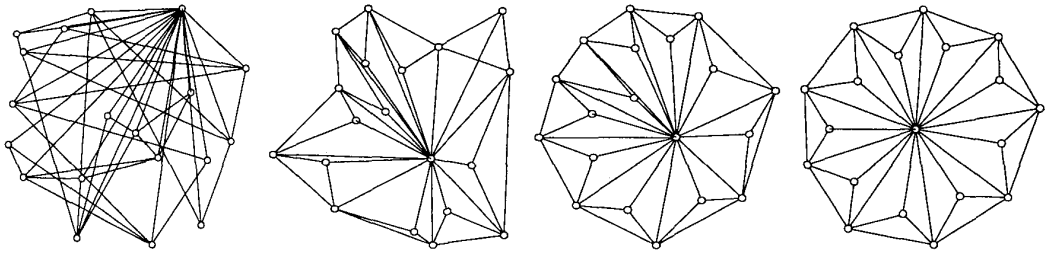
In Figure 2(a), just the En criterion was used. It is a good example of the robustness of this parameter. However, the symmetry of the pictures has a clear effect: the drawing has five edge crossings. When using the Ec parameter (Figure 2(b)), the result is not so good: the edges are far enough from the center of the picture just to avoid the crossings. To improve the result, the VEd criterion was used, which resulted in the improved Figure 2(c).

Another example of the effect of using different criteria can be seen when we try to layout a regular hexahedron (cube) graph. Figure 3 shows two pictures of this graph.

The well known k_{33} graph is presented in Figure 4. It is important to remember that it is a nonplanar graph. Davidson and Harel proposed a layout very similar to that presented in Figures 4(a) and 4(b). They noticed that the ideal picture was that in Figure 4(c), which we were able to find using the criteria and weights shown below the picture.

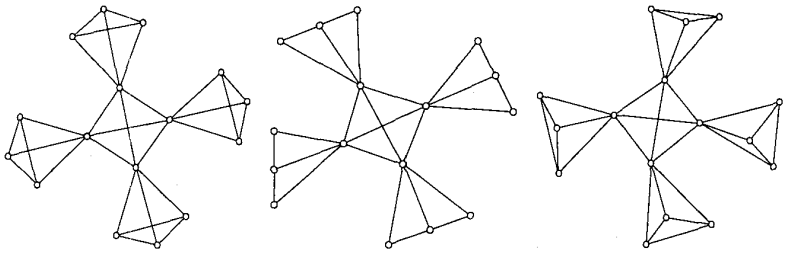
The next example shows new features of our algorithm. It concerns the regular dodechaedron, and was first introduced in [10]. Kamada and Kawai have drawn it exactly as in Figure 5(a). Just like them, we used the En criterion alone to lay it out. This graph can be also found in [5], [4] and [9]. All of them proposed a layout very similar to that presented in Figure 5(b). The interesting point is that it was obtained in our case, using the same criterion En, by prematurely interrupting the process. Davidson and Harel found a planar version of this graph. It can be seen in Figure 5(c). Here, we first used just the Ec parameter and after finding a planar (but not so nice) picture of it, we introduced the other criteria to lay it out.

Davidson and Harel suggested an ideal layout of the Heywood graph (Figure 6(a)), but were unable to produce it. They found, as Fruchterman and Reingold did, poor versions of it. The best picture of this graph until now was obtained by Coleman and Parker, which we were able to reproduce in Figure 6(b).



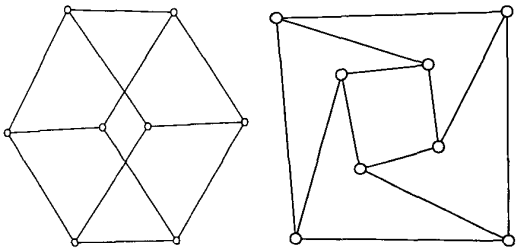
(a) Input graph (b) After 100 generations (c) After 200 generations (d) After 500 generations

Figure 1. The algorithm in action



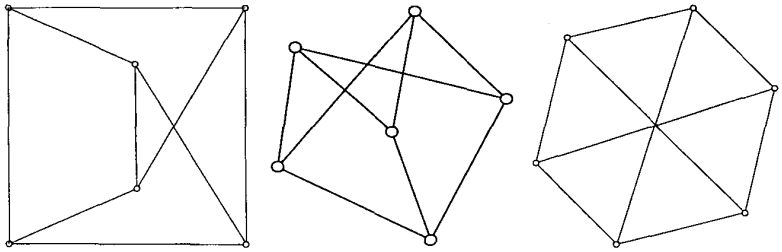
(a) $En \neq 0$ (b) $En = .02$, $Ec = .3$ (c) $En = .02$, $Ec = .3$, $VEd = .04$

Figure 2. Graph in Figure 3(c) from [10]



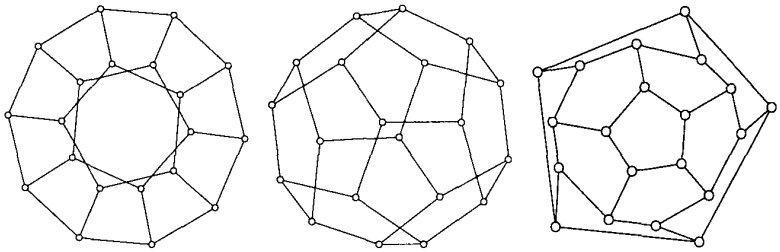
(a) $En \neq 0$ (b) $Ec = .14$, $VEd = .65$

Figure 3. Graph in Figure 3(a) from [10]



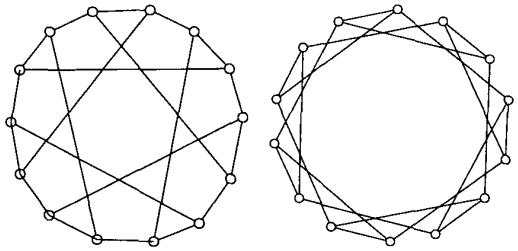
(a) $Ec = .02, VEd = 1$ (b) $En = .02, Vr = .06, VEd = 1$ (c) $Vr = .04, Cu = .02$

Figure 4. Graph in Figure 7 from [5]



(a) $En \neq 0$ (b) $En \neq 0$ (c) $Ec = 1, Ed = .02, Vr = .1, Ea = .1, VEd = .02$

Figure 5. Graph in Figure 3(d) from [10]



(a) Layout proposed by Davidson and Harel – *this layout was not obtained by our algorithm* (b) $En = .14, Ca = .08, Vr = .26, VEd = .3$

Figure 6. Graph in figures 21 and 22, from [9]

5. Conclusions and research in progress

We have proposed an algorithm to draw general undirected graphs for human understanding. The primary advantage of our algorithm is flexibility: we use concepts from two distinct approaches to the graph drawing problem, the force-directed placement technique and the aesthetic criteria optimization. In addition, there is the possibility of adjusting the aesthetic criteria's weights "on the fly", making it possible to achieve an optimal graph layout concerning a particular criterion and then improving it by changing the weights of the other criteria. This last aspect of the algorithm makes it difficult to compare performance. However, we consider speed the greatest weakness of our algorithm (as well as all the others which use GAs or simulated annealing). To address this issue, we are developing a more "intelligent" mutation, using Newton-Raphson's method to optimize the layout. Furthermore, as mentioned in Section 3, another work in progress is developing a crossover operator which avoids (or at least minimizes) the competing conventions problem.

In the near future, we intend to develop a true multi-objective formulation of the problem using a generational[6] or a non-generational[2] GA. With a true multi-objective formulation the user is relieved from the task of providing a good weight for each aesthetic criterion. The solution of the problem would no longer be the best ranking element in the population but the whole population itself which is evolved in order to approximate the Pareto set of non-dominated solutions, that is, those solutions where any improvement in a given criterion is only possible at the expense of degrading at least one other criterion.

Another research in progress, based in [13], is the evolution (by a GA) of an ideal set of weights for the criteria – reflecting the aesthetic preferences of the user – by learning from examples. Those weights would then be used by the GA to lay out graphs which hopefully will be more likely to please such user.

References

[1] G. D. Battista, P. Eades, R. Tamassia, and I. Tollis. Annotated bibliography on graph drawing algorithms. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.

[2] C. Borges and H. Barbosa. A non-generational genetic algorithm for multiobjective optimization. In *2000 Congress on Evolutionary Computation*, pages 172–179, San Diego, CA, USA, July 2000. IEEE Service Center.

[3] J. Branke, F. Bucher, and H. Schmeck. A genetic algorithm for drawing undirected graphs. In *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, pages 193–206, 1997.

[4] M. Coleman and D. Parker. Aesthetics-based graph layout for human comsuption. *Software Practice & Experience*, 26(1):1–25, 1996.

[5] R. Davidson and D. Harel. Drawing graphics nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, 1996.

[6] R. de Castro and H. Barbosa. A genetic algorithm for multiobjective structural optimization. In *IV Simpósio Mineiro de Mecânica Computacional*, pages 219–226, Uberlândia, MG, Brasil, May 2000.

[7] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[8] T. Eloranta and E. Makinen. TimGA - a genetic algorithm for drawing undirected graphs. Technical report, Department of Computer Science, University of Tampere, December 1996.

[9] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software Practice & Experience*, 21(11):1129–1164, 1991.

[10] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, April 1989.

[11] S. Kirkpatrick, C. Gellat, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[12] A. Kumar and R. H. Fowler. A spring modelling algorithm to position nodes of an undirected graph in three dimensions. Technical report, University of Texas-Pan American, Department of Computer Science.

[13] T. Masui. Evolutionary learning of graph layout constraints from examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pages 103–108. ACM Press, November 1994.

[14] N. Quinn and M. Breuer. A force directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, 26(6):377–388, 1979.