DRAWING GRAPHS USING SIMULATED ANNEALING AND GRADIENT DESCENT

Janez Brank
Department of Knowledge Technologies
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 1 4773778; fax: +386 1 4251038
e-mail: janez.brank@ijs.si

ABSTRACT

This paper presents graph drawing as an optimization problem. Each vertex of the graph is to be represented by a point in the plane, and each edge by a straight line between two points. To evaluate a drawing, an energy function is defined that depends on the coordinates of all the vertices. To find a good drawing, various optimization techniques, such as simulated annealing, can be used. We show a well-known example of an energy function and describe how it can be modified to become differentiable and thus suitable for minimization using gradient descent. We compare the results of this approach with the results of simulated annealing on several graphs.

1 INTRODUCTION

Graphs are one of the fundamental mathematical structures, interesting both for their theoretical aspect and for their usefulness in modeling many real-world phenomena. A graph is usually defined as $G = (V, E)$, where V is a set of vertices and $E \subseteq V^2$ is a set of edges. Depending on the application, the vertices and edges can also have additional attributes, such as length, capacity, color, and so on. Edges may be directed or undirected.

It is often desirable to represent a graph by a planar drawing, e.g. for display on paper or on a computer screen. This leads to the problem of graph drawing. We will consider the simplest form of graph drawing, in which each vertex is to be mapped to a point in the two-dimensional Euclidean plane, and each edge is represented by a line segment connecting the points into which the vertices connected by this edge have been mapped. (We note in passing that this problem can be generalized in many ways, for example by allowing edges to be represented by broken lines or by curves, or by considering representations in vector spaces of more than two dimensions.)

Under these assumptions the drawing of a graph is uniquely determined by the coordinates of the points representing its vertices. Let the vertices be numbered 1, 2, ..., n, where $n = |V|$, and let (x_i, y_i) be the coordinates of the point representing vertex i. The problem of drawing the graph has thus been reduced to selecting the values $x_1, y_1, \ldots, x_n, y_n$, typically under some constraints e.g. that $0 \le x_i \le 1$ and $0 \le y_i \le 1$.

One approach to choosing the coordinates is to define a criterion function $f(x_1, y_1, \ldots, x_n, y_n)$, the value of which should be in some sense related to the attractiveness of the resulting drawing of the graph, or its clarity, or whatever other criterion we are interested in. If smaller values of f are meant to imply a more attractive drawing, f will need to be minimized and is often called the *energy function*.

The relationship between f and the variables x_i, y_i, $i = 1, \ldots, n$, is often quite complicated. Therefore, the minimization of f is typically done using algorithms that make few if any assumptions regarding f, e. g. local optimization, simulated annealing, genetic algorithms, and so on.

In this work we will consider an energy function proposed by Davidson and Harel [1], which they minimized using simulated annealing. We will show how this energy function can be slightly modified to become differentiable, and its partial derivatives can then be computed and gradient descent can be used instead of simulated annealing to find a representation of the graph.

2 AN ENERGY FUNCTION FOR GRAPH DRAWING

The energy function proposed by Davidson and Harel [1] is a weighted sum of several components, each of which is intended to represent some criterion regarding the esthetic value of the drawing of a graph. Thus, we define $E = \lambda_1 E_1 + \ldots + \lambda_5 E_5$, where $\lambda_1, \ldots, \lambda_5$ are constant weights (their values being fixed before optimization begins), and $E_1, \ldots, E_5$ are the individual components that will be described below.

The first criterion is to prefer drawings in which the vertices are spaced evenly around the drawing area (we will assume that the coordinates of the vertices are constrained to lie in the unit square, i.e. $0 \le x_i \le 1$ and $0 \le y_i \le 1$). One way to encourage this is to have the vertices repel each other, thus preventing the algorithm from locating two or more vertices too close together. Thus, we define $E_1 = \Sigma_{i,j=1..n} 1/d_{ij}^2$, where $d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$ is the Euclidean distance between the vertices i and j.

Since the vertices repel each other, but are required to lie within the unit square, the criterion f_1 would by itself encourage the vertices to move near the edges of the unit rectangle, to be as far from each other as possible. However, the drawing of the graph tends to look nicer if the vertices are not too close to the edges of the unit square. Thus, we introduce a term which causes the edges of the unit square to repel the vertices, just as the vertices repel each other: $E_2 = \Sigma_{i,j=1..n} (1/x_i^2 + 1/y_i^2 + 1/(1-x_i)^2 + 1/(1-y_i)^2)$.

The criteria E_1 and E_2 introduced so far work towards having the vertices nicely spaced around the unit square but do not take the structure of the graph, i.e. its edges, into account. In our drawing of the graph, the two endpoints of each edge will be connected with a straight line. Thus the drawing is likely to appear less cluttered and have fewer edge crossings (which are a major difficulty to the viewer who is trying to understand the structure of the graph) if pairs of vertices that are connected by an edge lie closer together: $E_3 = \Sigma_{(i,j)\in E} d_{ij}^2$.

However, the term E_3, although it encourages the endpoints of edges to lie closer together, is not by itself sufficient to really discourage edge crossings. Thus, it is desirable to include the minimization of edge crossings explicitly as a fourth criterion: $E_4 = \Sigma_{e,f\in E} s_{ef}^2$, where s_{ef} is defined as 1 if the edges e and f intersect (unless the "intersection" is actually a vertex that is an endpoint of both e and f) and 0 otherwise.

Additionally, the clarity of the drawing suffers if a vertex is too close to an edge of which it is not an endpoint, as the viewer may be unsure whether the vertex is meant to lie on the edge or not. Thus we introduce a fifth criterion, $E_5 = \Sigma_{(i,j)\in E,\ k\in V-\{i,j\}} 1/\max(d_0, d_{ijk})^2$. Here d_{ijk} is defined as the distance of vertex k from the straight line connecting the vertices i and j; d0 is a constant (to be chosen before optimization begins) whose role is mainly to prevent division by 0 or excessively large values of E_5 in cases where a vertex lies very close to an edge.

3 SIMULATED ANNEALING

Simulated annealing is a well-known optimization technique. It is based on the analogy with annealing in physics, where the changes of the system state are seen as essentially random, but changes that reduce the energy are more likely than those that increase it. In addition changes that increase the energy are more likely while the temperature is high than later when the temperature is low.

These principles can be used to guide simulated annealing as an optimization technique in the following way. Let $E(\mathbf{u})$ be the function we are trying to minimize, where $\mathbf{u}$ is the vector of independent variables. In each step of simulated annealing, the algorithm considers making some small random step away from $\mathbf{u}$, into some new state $\mathbf{u}'$. If $E(\mathbf{u}') < E(\mathbf{u})$, i.e. the new state is better than the old one, the move is accepted and $\mathbf{u}'$ becomes the new current state. However, if $E(\mathbf{u}') \geq E(\mathbf{u})$, the move would increase the energy of the system, and is accepted only with probability $e^{-(E(\mathbf{u}')-E(\mathbf{u}))/T}$. That is, the greater the increase of energy, the less likely is that such a move would be accepted. If the move is rejected, $\mathbf{u}$ remains the current state and a new random step will be considered.

The role of T in the acceptance probability formula, $e^{-(E(\mathbf{u}')-E(\mathbf{u}))/T}$, is twofold. Firstly, it must bring the exponents into a reasonable range of values so as to result in reasonable acceptance probabilities. If T is too small, the acceptance probability will be very close to 0 and our search through the state space will be reduced to a greedy local optimization that only accepts moves which reduce energy. If T is too large, the acceptance probability will be close to 1, meaning that almost all moves will be accepted, even if they cause a large increase in energy; our algorithm will degrade into a random walk through the state space, ignoring to the energy function E. The second aspect of T is that it can be modified as the algorithm progresses. Typically, higher values are used initially to allow the algorithm to explore the state space using random jumps; slowly, the value of T is decreased, to encourage the algorithm to "settle down" in some low-energy area and end up in some local minimum. A period of pure local optimization might also be performed at the end.

The algorithm can be terminated after a certain number of moves or if the energy has not decreased sufficiently over a certain period of time.

In the case of graph drawing, our state consists of the coordinates of all the vertices, i.e. $\mathbf{u} = (x_1, y_1, \ldots, x_n, y_n)$. We followed the recommendation of Davidson and Harel and considered steps that move one of the vertices by a certain distance in a random direction.

Although it is a successful algorithm, simulated annealing has some drawbacks. In particular, the range of suitable values of T depends on the typical range of values of $E(\mathbf{u}')-E(\mathbf{u})$, which in turn depends on the graph that we are trying to draw (e.g. a larger graph has more vertices and edges and thus larger values of E). Thus, the initial value of T has to be chosen (based on a bit of experimentation) separately for each graph. Additionally, a scenario for decreasing the value of T over time has to be devised, where there are again many choices to be made. Typically a certain number of steps are made with a particular T, after which T is multiplied by 0.95 or some similar value. The need for making these choices and tuning these parameters motivated us to try using gradient descent as an alternative to simulated annealing for the optimization problem of graph drawing.

4 GRADIENT DESCENT

If the energy function $E(u_1, \ldots, u_m)$ is partially differentiable with respect to all the independent variables $u_1, \ldots, u_m$, we can compute its gradient vector $\nabla E = (\partial E/\partial u_1, \ldots, \partial E/\partial u_m)$. This vector, if computed in a point $\mathbf{u}$, is the direction in which we must move away from $\mathbf{u}$ if we want the function E

to increase as quickly as possible. Thus, to minimize the function E, we should make a step in the opposite direction. This brings us into a new point $\mathbf{u}' = \mathbf{u} + \lambda\nabla E(\mathbf{u})$, where λ determines the length of this step. We can stop after a certain number of steps or when the norm $\|\nabla E\|$ becomes sufficiently close to 0, suggesting that we reached a local minimum.

The challenge of using gradient descent in our graph drawing problem lies mainly in two aspects. Firstly, to take derivatives of E with respect to the coordinates x_i and y_i (which are our only independent variables), we must express E explicitly in terms of the coordinates, which in turn means expressing terms such as s_{ef} and d_{ijk} (see Section 2) as functions of the coordinates. Secondly, the energy function E as it has been defined in section 2 has many discontinuities where it is not differentiable. For example, the term s_{ef}, which indicates whether the edges e and f intersect, has a sudden (discontinuous) transition from 0 to 1 (or vice versa) where the two edges begin (or cease) to intersect, while everwhere else it is constant and thus its derivative is 0 and will not contribute anything towards the gradient (for example, it will not suggest how the coordinates of vertices should be moved to avoid a crossing). Similarly, the term $\max(d_0, d_{ijk})$, which occurs in E_5, is not differentiable at the transition where $d_{ijk} = d_0$.

First we show how to express d_{ijk} in terms of the coordinates of the vertices i, j, k. Let $\mathbf{r}_i = (x_i, y_i)$, $\mathbf{r}_j = (x_j, y_j)$, $\mathbf{r}_k = (x_k, y_k)$, $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, $\mathbf{r}_{ik} = \mathbf{r}_k - \mathbf{r}_i$. Now $\mathbf{r}_{ik}$ can be decomposed into a component parallel to $\mathbf{r}_{ij}$ and one orthogonal to it: $\mathbf{r}_{ik} = \nu\,\mathbf{r}_{ij} + \mathbf{r}'$, where $\mathbf{r}' \perp \mathbf{r}_{ij}$ (and thus $\mathbf{r}_{ij}^{\mathrm{T}}\mathbf{r}' = 0$). Taking the dot product of both sides by $\mathbf{r}_{ij}$ yields a formula for ν: $\nu = \mathbf{r}_{ij}^{\mathrm{T}}\mathbf{r}_{ik}/\|\mathbf{r}_{ij}\|^2$. Then $\|\mathbf{r}'\| = \|\mathbf{r}_{ik} - \nu\mathbf{r}_{ij}\|$ is the distance of vertex k from the straight line passing through vertices i and j. However, since we are only interested in the distance of k from the straight line segment from i to j, we must use $f_{01}(\nu)$ instead of ν, where $f_{01}(t) = \max(0, \min(t, 1))$.

A similar approach can be used for s_{ef}. Let $e = (i,j)$ and $f = (k,l)$. The straight line segment connecting i and j can be parameterized as $\mathbf{r} = \mathbf{r}_i + \lambda\mathbf{r}_{ij}$ for $0 \le \lambda \le 1$, and the line segment from k to l as $\mathbf{r} = \mathbf{r}_k + \mu\mathbf{r}_{kl}$ for $0 \le \mu \le 1$. The intersection of the two segments, if it exists, must satisfy both equations at the same time. Solving this system of two linear equations in two variables yields the formulas

$$\lambda = x_{ik}/x_{ij} - (x_{lk}y_{ik}/y_{ij} - x_{lk}x_{ik}/x_{ij})/(x_{ij}y_{lk}/y_{ij} - x_{lk}),$$

$$\mu = (x_{ij}y_{ik}/y_{ij} - x_{ik})/(x_{ij}y_{lk}/y_{ij} - x_{lk}),$$

where $x_{ij} = x_j - x_i$ and so on. The inequalities $0 \le \lambda \le 1$ and $0 \le \mu \le 1$ can be taking into account by defining $s_{ef} = v_{0,1}(\lambda)v_{0,1}(\mu)$, where $v_{a,b}(t) = 1$ if $a \le t \le b$ and 0 otherwise.

To ensure differentiability of E, we must now replace the functions $f_{01}(t)$, $v_{0,1}(t)$ and $\max(d_0, t)$ with some suitable differentiable approximations (see Figure 1). We will use approximations based on the sigmoid function $1/(1 + e^{-A(t-B)})$, which (assuming $A > 0$) has a value of approximately 0 when $t < B$ and approximately 1 when $t > B$. There is a smooth transition around $t = B$, the steepness of which depends on A. This function is continuous and differentiable for all t.

Recall that $f_{01}(t) = t$ for $0 \le t \le 1$, and it equals 0 for $t < 0$ and 1 for $t > 1$. We can approximate it by a sigmoid with a transition ½, i.e. by the function $1/(1 + e^{-A(t-1/2)})$. The integral of the squared approximation error is smallest when $A \approx 5.36$.

Similarly $v_{a,b}(t)$, which should be 1 for $a \le t \le b$ and 0 elsewhere, can be approximated by a difference of two sigmoids: $\underline{v}_{a,b}(t) = 1/(1 + e^{-A(t-a)}) - 1/(1 + e^{-A(t-b)})$. We could use $\underline{v}_{0,1}(t)$ to approximate $v_{0,1}(t)$, but this has the problem $\underline{v}_{0,1}(t) = ½$ for $t=0$ and $t=1$, but from the way that $v_{0,1}$ is used in the formula for s_{ef} we see that this would correspond to situations when an endpoint of one edge lies on some other edge. This is at least as undesirable as a general intersection where both λ and μ are somewhere between 0 and 1; thus, it is unfortunate that $\underline{v}_{0,1}$ has the value ½ rather than 1. Therefore, we will use $\underline{v}_{-\varepsilon,1+\varepsilon}$ (for some small positive ε, e.g. $\varepsilon = 0.1$) instead of $\underline{v}_{0,1}$.

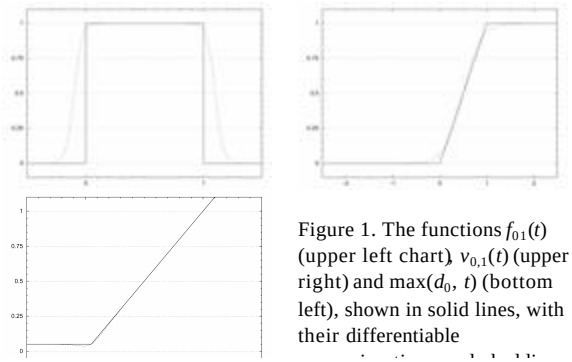

Figure 1. The functions $f_{01}(t)$ (upper left chart), $v_{0,1}(t)$ (upper right) and $\max(d_0, t)$ (bottom left), shown in solid lines, with their differentiable

Finally, using the same principles, $\max(d_0, t)$ can be approximated by $d_0 + (t - d_0)/(1 + e^{-A(t-d0)})$.

After these adjustments to the definition of the energy function E, its partial derivatives can be determined analytically using the well-known chain rule. As this derivation is somewhat tedious and not particularly illuminating, it will be omitted here and the interested reader is referred to [2].

5 EXPERIMENTAL EVALUATION

We compared simulated annealing (SA) and gradient descent (GD) on several graphs, shown in Figure 2. Our main evaluation measure is the number of edge crossings, as the experiments have shown that of all parts of the energy function this is the most difficult to minimize and that the appearance of the drawing often does not improve much if the other parts of the energy function are decreased while the number of crossings remains unchanged. We ran each technique (SA and GD) ten times, with different random initial configurations.

We also experimented with a "jumping heuristic" (JH) that can be used to further reduce the number of crossings after SA or GD have done their work. If edges ij and kl

intersect, this crossing can be prevented by moving i sufficiently close towards j (unless j lies on kl), or even by allowing i to "jump" over j. That is, the coordinates of i are moved from $\mathbf{r}_i$ to $\mathbf{r}_i + \lambda(\mathbf{r}_j - \mathbf{r}_i)$ for some $\lambda > 0$, possibly $\lambda > 1$. Of course, while this removes the crossing of ij and kl, it might produce one or more new crossings with other edges. Our heuristic tests several values of i and λ to see which yields the greatest decrease in the number of crossings.

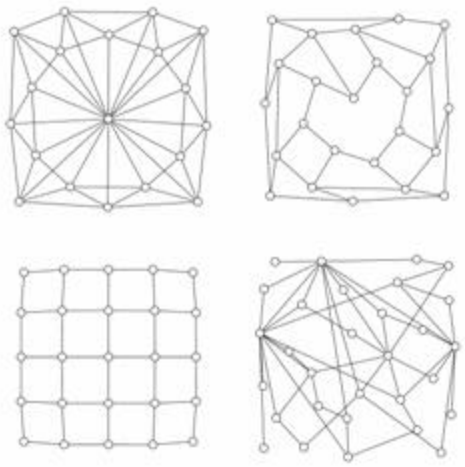

Figure 2. Four graphs used in our experiments.

The upper left graph of Figure 2 consists of 19 vertices and 54 edges. It is not planar, but can be drawn reasonably nicely with 9 crossings. All methods managed to find such a drawing at least once. On average, SA is better than GD, but the results of the latter are more amenable to improvmement with the JH and thus GD + JH is better than SA + JH on this graph.

The upper right graph of Figure 2 consists of three cycles of eight edges, connected into a kind of mesh for a total of 24 vertices and 40 edges. Here SA never found a crossing-free representation (SA + JH succeeed in one of ten attempts), while GD managed to do this in four out of ten runs. Davidson and Harel [1] also discuss this graph and managed to find a planar representation with SA, which probably suggests that they chose the annealing parameters differently. This underlines the sensitivity of SA to the choice of parameters, which is one of its main drawbacks.

The lower left graph is a rectangular mesh of 5×5 vertices and 40 edges. Here SA is more successful than GD, finding a planar representation in two out of ten attempts (GD: one out of ten), or in six out of ten with JH (GD: three out of ten).

Finally, the lower right graph is a random graph of 30 vertices and 50 edges, generated based on the principle of preferential attachment that is popular in modeling graphs with real-world properties similar to those of the web and various social networks [3]. As the graph is relatively dense and tangled, none of the techniques considered here found a representation with a really small number of edge-crossings. On average SA and GD perform comparably well, with GD again being more amenable to improvement with the JH.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented graph drawing as an optimization problem, based on an energy function introduced by Davidson and Harel. We showed how this function, originally intended to be minimized using simulated annealing, can with slight adjustments be made amenable to partial differentiation and minimization using gradient descent. Our experimental evaluation shows that both methods achieve comparable results and each has its own positive and negative sides. In the case of simulated annealing, it is necessary to choose parameters such as T (initial "temperature") and determine a cooling schedule. In the case of gradient descent, it is necessary (but not difficult) to choose a reasonable minimum length of each step to prevent the algorithm from falling into a local minimum too early.

The energy function considered here also has the drawback of being too computationally expensive to be useful for larger graphs. In general, the drawing of larger graphs needs to be based on different principles than those used here; the drawing must inevitably look too cluttered if each vertex is represented by a point and each edge by a straight line between two points. In such cases it would be better to first define a "frame", i.e. a smaller graph obtained by deleting and merging vertices of the original graph; after drawing the frame using an energy function such as the one considered here, the drawing could be augmented (perhaps at user's request, by a "zooming in" operation) by adding the vertices that were previously deleted to produce the frame. Similarly, vertices with a degree of 1 could be removed before drawing and later easily added without causing any new crossings.

Acknowledgments

I would like to thank Prof. Tomaž Pisanski for encouraging me to work on graph drawing, and Dunja Mladenic and Marko Grobelnik for encouraging me to write up the results in this paper.

References

[1] R. Davidson, D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[2] J. Brank. *A comparison of a few optimization algorithms for graph drawing*. Unpublished manuscript, April 2004. In Slovenian.

[3] A.-L. Barabási, R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 15 October 1999.