Journal of Computer Science & Its Applications, June 2006, Vol. 12 No.1

Drawing Graphs with Modified Simulated Annealing Algorithm

Adewole A., Philip
Department of Mathematical Sciences
University of Agriculture, Abeokuta.

ABSTRACT

This paper illustrates a modified simulated annealing algorithm for drawing graphs according to a number of aesthetic criteria. The proposed modified algorithm used in this paper combined the power of simulated annealing algorithm and mouse events available in java to enhance graph readability. Tests are carried out on two graphs of increasing difficulty, and the results show that this approach draws graph nicely and at the same time meets some the aesthetic criteria stated in this work.

Keywords: Aesthetic Criteria, Graph, Mouse Events, Simulated Annealing

1. INTRODUCTION

Graphs are one of the most useful structures in Science and Mathematics. A graph G = (N,E) is a set V of nodes (vertices) E is a set of ordered pairs of nodes called arcs (edges). Graph drawing is a hard problem and therefore, exact solution methods are only feasible for small or greatly restricted problems.

A good number of data representation problems involve the drawing of a graph on a two dimensional surface, like a sheet of paper or a computer screen. Examples include circuit schematics, communication and public transportation networks, social relationships and software engineering diagrams. Mostly, in all presentation applications, the usefulness of a graph drawing depends on its readability. This refers to the ability of the graph to convey the meaning of the diagram quickly and clearly. Readability issues are expressed by means of aesthetics, which can be formulated as optimization criteria for the drawing algorithm [1].

A detailed aesthetic criteria that have been proposed and various heuristic methods for satisfying them can be found in [2]. The methods proposed vary according to the class of graphs for which they are intended and the aesthetic criteria they take into account. In general, the optimization problems associated with most aesthetic are NP–hard [6]. For instance, it has been proven that even just minimizing the number of edge crossings is an NP–hard problem [6].

Given a partially connected graph G = (V,E), we wish to determine the coordinates for all vertices in V on a plane so as to satisfy a certain number of aesthetic criteria. This paper proposes the use of a modified simulated annealing algorithm for drawing graphs based on the following aesthetic criteria.

- Avoid edge crossings
- Keep edge lengths uniform
- Distribute vertices uniformly
- Minimize aspect ratio
- Minimize area
- Minimize total edge length
- Minimize maximum edge length

:

The proposed modified algorithm used in this paper combined the power of simulated annealing algorithm and mouse events provided in java to enhance graph readability. Tests are carried out on two graphs of increasing difficulty, and the results show that this approach draws graph nicely and at the same time meeting the aesthetic requirements used in this work.

2.0 RELATED WORK

Several algorithms have been developed and used in drawing graphs. Spring model version proposed by [4] works by replacing edges with springs unit of some natural length. In his approach, vertices are initially placed randomly and a system of different equations is solved to find the system state with minimum energy. By nature, springs attract their endpoints when stretched and repel their endpoints when compressed. Vertices that are adjacent are kept close to each other by shorter springs. While the longer springs between non – adjacent vertices kept them apart, and at the same time they limit the overall size of the embedding.

The most popular algorithm for laying out a (k,2)–partite graph was introduced by [8]. The algorithm takes an hierarchical directed graph as input and draws it in four stages. In stage I, the graph is transformed into a "proper" hierarchy, if necessary. In stage II the vertices at each level are ordered to reduce the number of edge crossings. In stage III, the horizontal position of each vertex is manipulated to reduce the length of the edges while in stage IV the graph is drawn. [3] proposed a spring-embedding approach that unlike previous approaches results in a model that is convex while laying out a graph. [4] introduced techniques for drawing graphs based on force-directed placement and virtual physical models in producing good layouts of undirected graphs.

The shortcoming with these algorithms is their inability to permit user to modify graph drawn so as to enhance graph readability. This paper proposes the use of a modified simulated annealing algorithm that enhances graph readability. With this approach user can through the use of mouse event facility available in java manually rearrange their graphs to make graph more readable.

3.0 SIMULATED ANNEALING

Simulated annealing is a well – known optimization technique. It is based on the process by which some substance is heated until it is a liquid and then slowly cooled until it is a solid. The slow cooling allows the molecules to organize themselves into a crystal, a totally ordered form. In other words 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 increased it. In addition changes that increase the energy are more likely while the temperature is high than when the temperature is low.

This process is often used in the manufacturing of products such as steel. Simulations of annealing were developed to analyze the efficiency of these manufacturing processes. Simulated annealing is used where the potential solution is large and a combinatorial search is infeasible.

The above-mentioned principles can be used to guide simulated annealing as an optimization technique. The simulating annealing pseudocode follows:

```
Initialize temperature T. follows:
Initialize temperature T;
        Initialize configuration $\beta$
        E $\leftarrow$ cost of $\beta$
While (min T not reached) OR (user specified constraint
not satisfied)
  While (termination condition not satisfied) choose a new
  configuration $\beta^1$ from the neighborhood of $\beta$;
  E$^1$ $\leftarrow$ cost of $\beta^1$
  If ( E$^1$ < E) OR (random < $e^{(E-E')/T)}$
  Then
  $\beta \leftarrow \beta^1$;
  E $\leftarrow$ E$^1$;
  End if
  Decrease temperature T
  End while
  Modify Layout with Mouse.
  End while
```

As can be seen from the pseudocode there are two major processes that are occurring during the simulated annealing algorithm. First, for each temperature the simulated annealing algorithm runs through a number of cycles. This number of cycle is predetermined by the programmer. Second, an important part of the simulated annealing process is how the inputs are randomized. This randomization process takes the previous values of the inputs and the current temperature as inputs. As the cycles runs the inputs configuration are randomized. In the case of graph drawing problem, these inputs are the coordinates of the points representing vertices. Only randomization which produce a better set of input is kept. Once the specified number of cycles has been completed, the temperature can be lowered. Once the temperature is lowered, it is determined if the temperature has reached the lowest allowed temperature. If the temperature is not lower than the lowest allowed temperature, then the temperature is lowered and another cycles of randomizations will take place. If the temperature is lower than the minimum temperature allowed, the simulated annealing algorithm is complete.

A neighborhood of a configuration β, is the set of configurations that differ from β^1 by the location of a single vertex. β^1 is generated by taking a vertex and placing it on a rectangle drawn around its original location. The cost of a configuration must be carefully chosen so that it reflects the desired aesthetics of the graph and is not overly computationally intensive to compute.

(i) In order to ensure that vertices are evenly distributed, the term

$E_1 = \Sigma^n_{1\,ij}\ \alpha_1/d^2_{ij}$ is added to the cost function pair of vertices, i,j. d^2_{ij} is the Euclidean straight line distance between i and j (i.e. $d^2_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2$ α_1 is a weighting factor whose value depend on the importance of this criteria relative to the others in the cost function.

(ii) A completely minimized cost function will spread out the vertices indefinitely. To make sure that vertices are drawn on the display area, the following term is added to the cost function for each node i. $E_2 = \Sigma^n{}_1 ij\ \alpha_2\ (1/r_i^2 + 1/l^2_i + 1/t_i + 1/b_i^2)$. Where r_i, l_i, t_i and b_i are the straight–line distances between the vertex and the right, left, top and bottom borders of the display area. Again α_2 is a weighting factor.

(iii) In order to avoid unnecessary long edges, the following term is added for each edge k of length d^2_k, with weighting factor α_3: $E_3 = \Sigma_{ij \in E}\ \alpha_3 d^2_k$.

(iv) To minimize the number of edge crossing $E_4 = \Sigma_{ef \in E} S^2_{ef} \alpha_4$ where S_{ef} is defined as 1 if the edges e and f intersect and o otherwise, α_4 is the weighting factor, is added to the cost function for pair of edges that cross.

(v) To prevent edges from being drawn too close together, particularly those that intersect at a vertex, the following term is added to every vertex i, edge k with distance jik and weighting factor α_5:
$E_5 = \Sigma_{(ij) \in E,\ k \in V-(ij)}\ 1/\max(d_0, d_{ijk})^2$. Here d_{ijk} is defined as the distance of vertex k from the straight line connecting the vertices i and j, d_0 is a constant (to be chosen before optimization begins) whose role is mainly to prevent division by o or excessively large values of E_5 in cases where a vertex lies very close to an edge.

Each criterion is intended to represent the aesthetic value of the drawing of a graph. Thus, we define $E = E_1 + E_2 + E_3 + E_4 + E_5$; $\alpha_1, \alpha_2, \ldots, \alpha_5$ are constant weights (their values being fixed before optimization begins), and $E_1, \ldots E_5$ are the individual components of the weighted sum. The relative importance of each criteria can be adjusted by varying the weighing factors.

For instance, let $E(\beta)$ be the function we are trying to minimize, where β is the vector of independent variables. In each step of simulated annealing, the algorithm considers making some small random step away from β, into some new state β^1. if $E(\beta^1) < E(\beta)$, i.e. the new state is better than the old one, the move is accepted and u^1 becomes the new current state. However, if $E(\beta^1) \geq E(\beta)$, the move would increase the energy of the system, and is acceptable only with probability $e^{-(E(\beta^1)-E(\beta)'))/T}$. That is, the greater the increase of energy, the less likely is that such a move is rejected, β remains the current state and a new random step will be considered.

The role of T in the acceptance probability formula, $e^{(E(u')-E*u))/T}$, is twofold. Firstly, it must bring the exponents in reasonable acceptance probabilities. If T is too small, the search through the state space will be reduced to a greedy local optimization that only accepts moves which 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 ignoring to the energy function E. The second aspect of T is that it can be modified as the algorithm progress. 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.

In particular, the range of suitable values of T depends on the typical range of values of $E(E^1) - E(u)$, 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.

The actual value of the initial temperature depends on how many iterations are desired in the annealing process. At each temperature level, approximately 30n perturbations should be performed. The cooling function should be geometric, i.e. $T_{p+1} = \gamma T_p$, with $0.6 \leq \gamma \leq 0.95$. a value of 0.75 gives a relatively rapid cooling with the resulting graphs giving good aesthetic properties. The termination condition can be fixed number of iterations, say 10, or when the proposed drawing stops changing significantly over two or three iterations.

Finally, if the user specified constraint is not satisfied the layout can then be modified manually by using mouse to rearrange the graph.

4.0 EXPERIMENTAL EVALUATION

The author compared simulated annealing and modified simulated annealing on two graphs, shown in Figures 1 and 2. The evaluation criterion used is the number of edge crossings: In [6] it has been established that in general, one cannot simultaneously optimize two aesthetic criteria. The results show that the proposed modified simulated annealing algorithm produces graph with minima number of edge crossings even in a complex graph while ordinary simulated annealing algorithm output is characterized with a huge number of edge crossings even for few number of nodes and edges. Figure 1 depicts the results produced by ordinary simulated annealing and the proposed modified simulated annealing for a graph with 30 vertices and 49 edges while figure 2 shows the corresponding results for a graph with 30 vertices and 79 edges. The program was implemented using Java programming language on Pentium II running Microsoft Windows 2000 Professional.

5.0 CONCLUSION AND FUTURE WORK

In this paper , a modified simulated annealing algorithm for drawing graph is presented. The algorithm was constructed by combining mouse event facilities available in Java with simulated annealing algorithm. With this modified simulated annealing it is possible to make graph more readable by allowing the user to move nodes to where the user felt it should be. To the best of our knowledge none of the previous graph drawing heuristics has employed this approach. However, it is the goal of the author to design a more powerful simulated annealing algorithm that will be able to improve graph readability without user intervention.

REFERENCES

[1] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. IEEE Transactions on systems, man and cybernetics, 18(1):61-79,1988.

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

[3] C. Ignacio, S. Thaddeus. A spring-embedding approach for the facility layout problem. Journal of the Operational Research Society, Vol.55, 73-81, 2004. *doi:10.1057/palgrave.jors.2601647*

[4] R. Emden, C. Stephen.Improved Force-Directed Layouts. http*://www. research att.com/info/ {erg.north}*

[5] Harmanani, H., Zouein, P., ASCE, A.M., and Hajar, A. An Evolutionary Algorithm for Solving the Geometrically Constrained Site Lay out problem Problem, Journal of Computer in Civ. Engrg., ASCE, 16(2), pp.331- 338, 2004.

[6] Nelson, W., Sheelagh, C., and Saul G. EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs, Journal of Graph Algorithms and Applications, 8(1),pp69-74,2004.

[7] Neville, C., Warwick, I., and Carl C. Inhomogeneous Force- Directed Layout Algorithms in the Visualization Pipeline: From Layouts to Visualization, Conferences in Research and Practice in Information Technology,35,pp.121-128,2004.

[8] Susan, S. Automatic Graph Drawing Algorithms, DIMACS, International Workshop, Springer Verlag, Berlin, pp1-15,1996.