DEPARTMENT OF MATHEMATICS
TECHNICAL REPORT

IMPROVING BIPARTITE GRAPHS USING MULTI-strategy SIMULATED ANNEALING

Michael Breen and Panagiotis K. Linos

April 1999

No. 1999-4

TENNESSEE TECHNOLOGICAL UNIVERSITY
Cookeville, TN 38505

Visualizing Bipartite Graphs Using Multi-strategy Simulated Annealing

Panagiotis K. Linos
Department of Computer Science
Tennessee Technological University
Cookeville, TN 38505
email: linos@tntech.edu

Michael Breen
Department of Mathematics
Tennessee Technological University
Cookeville, TN 38505
email: mbreen@tntech.edu

Abstract: The technique of simulated annealing combined with various greedy methods is applied to random bipartite graphs of different sizes and densities in order to reduce the number of edge crossings. Experimental results demonstrate that when random moves are combined with systematic steps within simulated annealing, the edge intersections in bipartite graphs can be reduced considerably in a reasonable amount of execution time. In particular, one of the cooling techniques used during the experiments reduces the execution time by half without sacrificing the quality of the layouts. Finally, we conclude by discussing work-in-progress of a prototype tool, which can facilitate the process of visualizing various layouts of bipartite graphs during the experiments.

1. Introduction

The problem of minimizing the number of edge intersections in bipartite graphs has been shown to be NP-complete [Garey 83]. Many greedy heuristics have been proposed and experimentally tested to find sub-optimal solutions within polynomial time [Junger 95, Linos 91, Eades 94, Battista 94]. One of the main disadvantages of the greedy heuristics is that they may get trapped in a local minimum which may be far worse than the global one, and have no way of leaving to that local minimum. As a possible remedy to this problem, the method of *simulated annealing* has been studied in order to address the general layout problem of graphs [Davidson 89]. This approach allows for more drastic steps to be taken, based on a probability, from a given state to a worse one in hopes of eventually arriving at or getting closer to the global minimum. Simulated annealing permits movements from a local minimum to help avoid getting trapped in an undesirable position. So far, experimental results demonstrate that simulated annealing can provide a promising approach, although slow, for producing aesthetically pleasing graph drawings [Coleman 96].The main objective of this work is to apply the technique of simulated annealing with a focus on minimizing edge intersections in bipartite graphs. More specifically, the combination of simulated annealing with various greedy methods and its effect on bipartite graphs is experimentally studied.

The rest of this paper is organized as follows: the second section presents some necessary terminology and the third presents the proposed *multi-strategy* simulated annealing method. The fourth section describes the experimental study, and the fifth one discusses some research issues that are currently under further investigation.

2. Terminology

A graph G(V,E) with a vertex set V and an edge set E is called a *bipartite* graph if V can be partitioned into two nonempty sets V_1 and V_2 such that $V = V_1 \cup V_2$ and $E \subseteq V_1 \times V_2$. The bipartite graphs considered in this paper all have the property that $|V_1| = |V_2|$, but that is not necessary in general. If $|V_1| = n$ then a bipartite graph can be drawn in the following manner. Each vertex is a point (i,b) in the plane where $1 \leq i \leq n$ and b = 0 or 1. Each edge is a line segment from (i,b_1) to (j,b_2) where $1 \leq i, j \leq n$ and b_1 and b_2 are either 0 or 1 but not equal. Two edges are said to cross or intersect if their line segments intersect at some point other than at a vertex. The set of vertices that have the same second coordinate is called a *level*. A bipartite graph can be represented by an n x n matrix in which the (i,j)-th entry is 1 if there is an edge from $i \in V_1$ to $j \in V_2$, and 0 if no such edge exists.

3. Multi-Strategy Simulated Annealing

The technique of *simulated annealing* has been applied to several optimization problems including the problem of producing aesthetically pleasing graph drawings [Davidson 89]. Some of the history of simulated annealing as well as its applications to other areas are described in [McLaughlin 89, Johnson 89]. In the case of minimizing edge intersections in bipartite graphs, if a heuristic rearranges a graph into one that has fewer intersections, the new graph is always accepted. An arrangement with more intersections is accepted whenever a randomly-generated probability taken from the uniform distribution between 0 and 1 is less than $e^{-\Delta/T}$ where Δ is the increase in the number of intersections and T is what is called the *temperature* of the system. So, if the increase in the number of intersections is very large, the new graph is unlikely to be accepted. Also, the higher the temperature, the more likely that a configuration with more intersections is accepted.

In this work, we present six different variations (or strategies) of simulated annealing which are summarized in Table I. In this table, each row contains a simulated annealing strategy in which two different layout methods are combined. For each strategy, the drastic step describes how vertices are rearranged when the temperature is decreased during the annealing process. The modest step shows how vertices are rearranged while the temperature is being held constant. The terms *Averaging, Greedy Adjacent Pair Switching* and *Greedy Insertion* used in Table I refer to three classic heuristics described in [Eades 89]. Briefly, we explain them here. In the *averaging* heuristic a vertex i is placed in position j, where j is the average of the positions of its adjacent vertices on the opposite level. The *greedy adjacent pair switching* heuristic considers consecutive vertices i and i+1 on the same level. The two vertices are switched. If the graph has fewer intersections after the switch, that new configuration is accepted. Otherwise, the vertices are returned to their original positions. The process of rearrangement continues on the same level until no consecutive switches yield improvement. The method is then applied to the opposite level. The *greedy insertion* method involves placing each vertex, one at a time, in each of the positions 1 through n on the same level and selecting the position for that vertex that gives the fewest number of intersections. The process of insertion is then performed on the opposite level. For all the above described heuristics, the overall process is repeated between levels until no further improvement is accomplished.

The rest of the heuristics follow a more random approach. Specifically, the *random pair switching* heuristic switches pairs of vertices randomly. The number of pairs to be switched is chosen at random as are the pairs themselves. The *random rearrangement* is a permutation, chosen at random, on all n vertices on a level. The *limited random insertion* involves choosing k random vertices on which to perform greedy insertion. In this case, both the number of vertices and the vertices themselves are chosen at random. Finally, the *random insertion* approach chooses the vertices as done in the *limited random insertion* but inserts each into a position chosen at random.

4. An Experimental Study

An experimental study is set up in order to study how the simulated annealing technique can improve the layouts of bipartite graphs. In particular, the main objective of the experiments is to compare the performance, with respect to the reduction of edge intersections, of the six different variations of simulated annealing described in the previous section.

4.1. Framework

Two performance metrics are used during the experiments; the *reduction of edge intersections* as a percentage of the original number of intersections and the *execution time* needed. The experiments are performed for different combinations of graph sizes and densities. The size n corresponds to the number of vertices on either level of the graph and the density represents the number of edges in the given graph divided by n^2. A summary of the experimental planning is shown in Table II. Specifically, we run experiments for bipartite graphs with size no greater than 100 vertices. For each size and density we randomly generate twenty different graphs.

TABLE I : *Description of multi-strategy simulated annealing*

	DRASTIC STEP	MODEST STEP
STRATEGY I	*Averaging*	*Greedy Adjacent Pair Switching*
STRATEGY II	*Averaging*	*Greedy Insertion*
STRATEGY III	*Random Pair Switching*	*Greedy Adjacent Pair Switching*
STRATEGY IV	*Random Rearrangement*	*Limited Random Insertion*
STRATEGY V	*Random Rearrangement*	*Random Pair Switching*
STRATEGY VI	*Random Rearrangement*	*Random Insertion*

TABLE II : *Summary of the experimental planning*

Metrics	1. Reduction of edge intersections as a percentage 1. Execution time
Input Parameters **Input Parameters, (contd.)**	1. Size = {10, 20, 30, 40, 50, 60, 80, 100} 2. Density = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6} 3. Initial temperature = {0.8} 4. Temperature length = {.5n} 5. Temperature cooling method = {geometric, standard deviation, symmetric swap} 6. Definition of *frozen* ={temperature=0.05}
Graphs Per Strategy	20 graphs x 48 (from 1 and 2 above) x 3 cooling methods = 2880 graphs
Total Experiments	6 heuristics x 2880 graphs = 17,280 graphs

For each graph, the overall reduction of edge intersections is measured and the average of the reduction of the twenty graphs is computed. Three decisions that must be made when implementing simulated annealing are how high the initial temperature should be, how the temperature should be reduced, and how low the temperature should be allowed to go before quitting. In our experiments, we set the temperature function to be initially high at 0.8 and then let it decrease gradually. When the temperature is initially high, it is more likely at the beginning of the experiments that a graph with more intersections will be accepted. All six simulated annealing strategies are tested using three different cooling methods. The first one is the *geometric* method in which the temperature is decreased by multiplying the previous temperature by .95. The second method is described in [McLaughlin 89] and it uses a *standard deviation* in order to determine the temperature decrease. In the *symmetric swap* method, the temperature is decreased by multiplying by one of three numbers. The multiplier is chosen based on the most recent improvement in the number of intersections. A large reduction in the number of intersections results in using the largest multiplier, so the system "cools" more. A small reduction leads to the use of the smallest multiplier so the system cools only a little. A reduction in the middle results in the use of the middle multiplier. In all three cooling methods, the simulated annealing process terminates when the temperature reaches 0.05. All six strategies are implemented in C++ and the experiments are performed on a dedicated high-speed PC platform. The implementation also includes a random generator for bipartite graphs.

4.2. Discussion of Particular Results when the Density is 0.4

In this section, we discuss some results of the experiments using the three cooling methods on graphs with density 0.4 and size from n = 10 to n = 60. In these experiments, we measure both the execution time (Figures 1-3), and the reduction of intersections as a percentage with respect to the initial number of crossings (Figures 4-6). Each point on the following six charts represents an average time/improvement of the twenty graphs experimented on for that particular size and cooling method. With regard to execution time, the average time taken by the *symmetric swap* cooling method is about half that of the other two cooling methods without any loss in improvement. Thus, it appears to be the best cooling method of the three tested. With regard to improvement, both the range and the shape of all improvement graphs for all cooling methods are very similar. For all three cooling methods, *Strategy IV* takes the longest amount of time and *Strategy II* takes the next longest. Also for all cooling methods experimented on, the other four strategies are clustered together, requiring the same amount of time. In each cooling method, *Strategy IV* takes about 2.5 times as long and *Strategy II* takes about 1.5 times as long as the other four strategies do, on average. Each improvement chart shows two strategies separate from the other four. The top two strategies are *Strategies I* and *III,* with improvement ranging from 30% for n = 10 to 11% for n = 60. Since both these strategies are in the group of four strategies having minimal time, *Strategies I* and *III* are the preferred strategies for all cooling methods. The other four strategies in decreasing order of improvement are *Strategy IV, Strategies V* and *VI* (very close together) and *Strategy II.* In each cooling method, the percentage of improvement of these four strategies converges to about 5% as the size of the graphs experimented on increases, but still maintains separation from the improvement of the top two strategies. Although *Strategy IV* is the closest to the top two strategies in improvement, it also has the greatest execution time. Since *Strategy II* is worst in improvement and takes the second greatest amount of time, it is the least desirable method. All of the data in this section refer to graphs with a density of 0.4, but the data are representative of all densities tested as evidenced in the next section.

4.3. Relative Comparison of the Experimental Results

Tables III through VIII on the following page summarize the average execution times and average per cent improvement for all six strategies and all three cooling methods for d = .1 through .4 and from n = 10 to n = 60. Each strategy is compared to *Strategy III* both in time and improvement. Each individual number in the table is the ratio of the time/improvement for that strategy to the time/improvement of *Strategy III* for a particular cooling method, density and size. The two numbers separated by a hyphen represent the lower and upper bounds for that ratio for the two densities and three sizes that determine the individual cell. For example, in Table III the entry 1.03-1.25 in the Size=20-30 and Density=0.3-0.4 cell for *Strategy I* means that *Strategy I* on the average, took at least 1.03 times as long and at most 1.25 times as long as *Strategy III* for those sizes and densities. The execution time ratios for graphs with 10 vertices per level have been omitted because the total execution time for the 20 graphs was less than 1.00 seconds. As with the particular density discussed in the previous section, *Strategies I* and *III* are very close to one another and while the execution times for both are similar to those for *Strategies V* and *VI, Strategies I* and *III* are better in improvement than all other strategies. That is, *Strategies I (averaging and greedy adjacent pair switching)* and *III (random pair switching and greedy adjacent pair switching)* are the best strategies when both time and improvement are taken into account. *Strategy IV (random rearrangement and limited random insertion)* is the slowest of the strategies and *Strategy II (averaging and greedy insertion)* is the least desirable of the six strategies.

5. Work in Progress

Currently, we are working towards various improvements of our simulated annealing strategies. These efforts include attempts to reduce the execution time by either reducing the initial temperature or in more drastic cases, terminating the whole process when insignificant improvement is made. Moreover, we are experimenting with smaller limits on the number of iterations and/or passes on each level of the bipartite graph. In addition, a prototype tool for visualizing layouts of bipartite graphs is under development. It is implemented using Tcl/Tk which is a programming environment that includes a scripting language for developing graphical user interfaces [Welch 95].

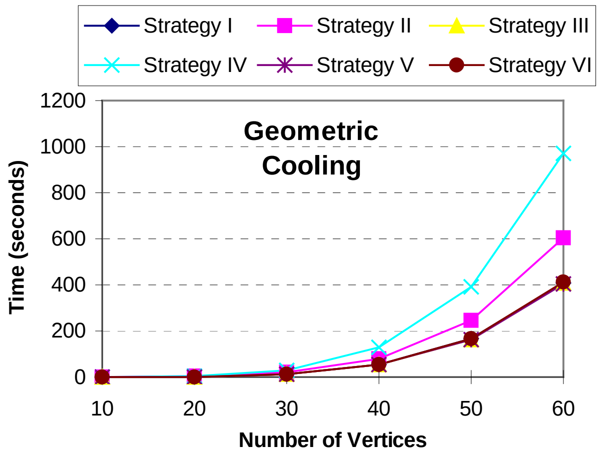

Figure 1: Average Execution Time Geometric Cooling method

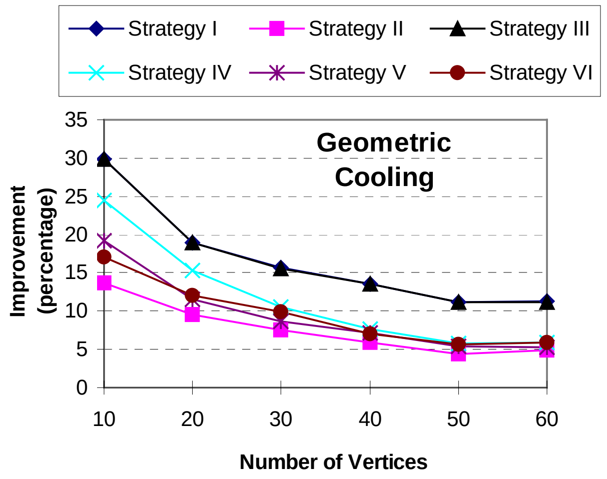

Figure 2: Average Improvement Geometric Cooling method

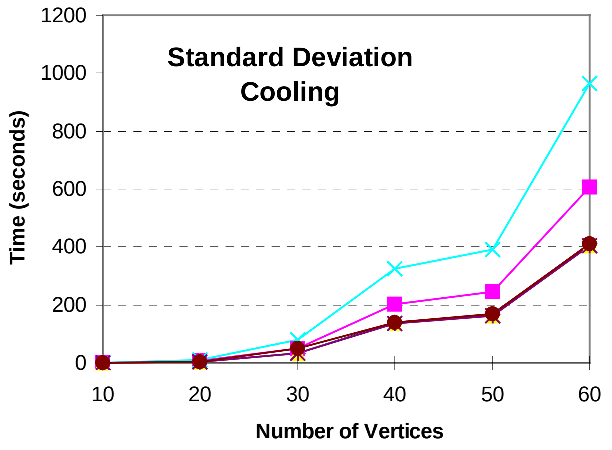

Figure 3: Average Execution Time Standard Deviation Cooling method

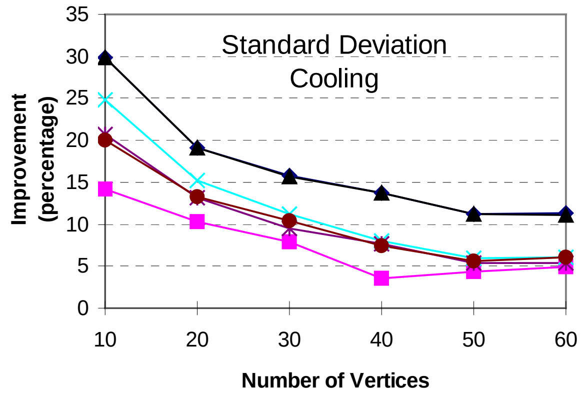

Figure 4: Average Improvement Standard Deviation Cooling method

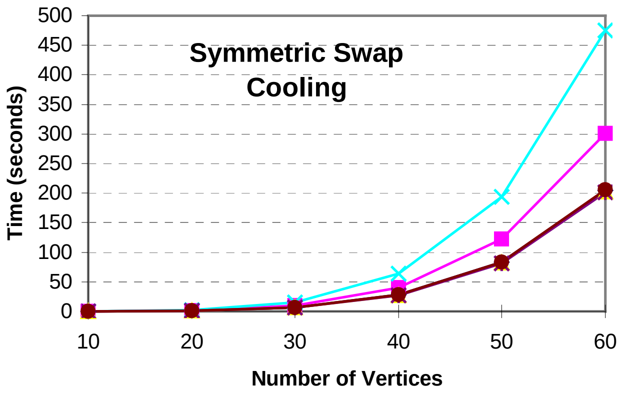

Figure 5: Average Execution Time Symmetric Swap Cooling method

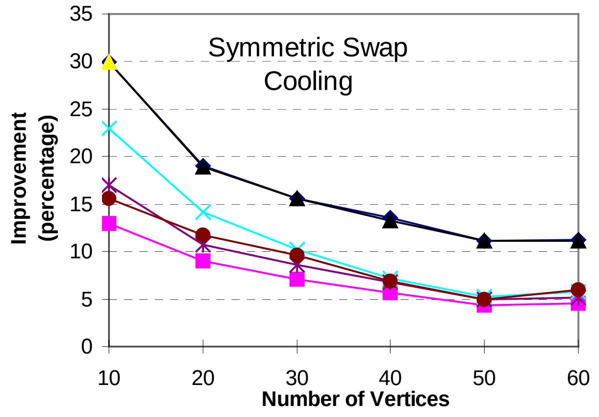

Figure 6: Average Improvement Symmetric Swap Cooling method

TABLE III : *Relative comparison of execution time for geometric cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=20-30	STRATEGY I	0.78-1.46	1.03-1.25
	STRATEGY II	1.43-1.77	1.40-1.63
	STRATEGY IV	2.56-3.15	2.24-2.71
	STRATEGY V	0.78-1.46	0.95-1.08
	STRATEGY VI	1.11-1.46	1.04-1.08
Size=40-60	STRATEGY I	1.03-1.07	1.01-1.02
	STRATEGY II	1.50-1.56	1.49-1.51
	STRATEGY IV	2.62-3.05	2.38-2.50
	STRATEGY V	1.00-1.07	1.00-1.01
	STRATEGY VI	1.07-1.24	1.01-1.07

TABLE IV : *Relative comparison of performance for geometric cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=10-30	STRATEGY I	0.89-1.00	1.00-1.00
	STRATEGY II	0.50-0.76	0.41-0.61
	STRATEGY IV	0.66-0.81	0.63-0.81
	STRATEGY V	0.58-0.77	0.53-0.74
	STRATEGY VI	0.58-0.74	0.57-0.73
Size=40-60	STRATEGY I	0.94-1.06	1.00-1.07
	STRATEGY II	0.35-0.52	0.40-0.49
	STRATEGY IV	0.45-0.66	0.52-0.61
	STRATEGY V	0.41-0.60	0.48-0.55
	STRATEGY VI	0.43-0.62	0.51-0.57

TABLE V : *Relative comparison of execution time for standard deviation cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=20-30	STRATEGY I	1.05-1.14	1.01-1.13
	STRATEGY II	1.24-1.55	1.44-1.49
	STRATEGY IV	2.63-3.06	2.37-2.55
	STRATEGY V	0.95-1.08	0.99-1.02
	STRATEGY VI	1.10-1.48	1.03-1.13
Size=40-60	STRATEGY I	1.02-1.08	1.01-1.03
	STRATEGY II	1.50-1.54	1.49-1.51
	STRATEGY IV	2.64-3.06	2.37-2.52
	STRATEGY V	1.00-1.06	0.99-1.00
	STRATEGY VI	1.07-1.27	1.02-1.06

TABLE VI : *Relative comparison of performance for standard deviation cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=10-30	STRATEGY I	0.92-1.00	0.99-1.00
	STRATEGY II	0.56-0.79	0.44-0.64
	STRATEGY IV	0.65-0.80	0.70-0.99
	STRATEGY V	0.60-0.86	0.57-0.76
	STRATEGY VI	0.58-0.77	0.61-0.76
Size=40-60	STRATEGY I	0.93-1.07	1.00-1.06
	STRATEGY II	0.35-0.54	0.40-0.50
	STRATEGY IV	0.43-0.67	0.53-0.58
	STRATEGY V	0.40-0.58	0.48-0.56
	STRATEGY VI	0.42-0.62	0.40-0.59

TABLE VII : *Relative comparison of execution time for symmetric swap cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=20-30	STRATEGY I	0.67-1.22	0.98-1.09
	STRATEGY II	0.83-1.63	1.37-1.67
	STRATEGY IV	1.50-3.17	2.26-2.92
	STRATEGY V	0.67-1.22	0.95-1.08
	STRATEGY VI	0.88-1.83	0.92-1.09
Size=40-60	STRATEGY I	1.02-1.08	1.01-1.02
	STRATEGY II	1.48-1.94	1.47-1.51
	STRATEGY IV	2.60-3.72	2.35-2.49
	STRATEGY V	1.00-1.31	0.98-1.01
	STRATEGY VI	1.07-1.51	1.02-1.07

TABLE VIII : *Relative comparison of performance for symmetric swap cooling*

		Density 0.1-0.2	**Density 0.3-0.4**
Size=10-30	STRATEGY I	0.91-1.01	0.99-1.00
	STRATEGY II	0.46-0.78	0.40-0.60
	STRATEGY IV	0.64-0.78	0.61-0.89
	STRATEGY V	0.54-0.83	0.51-0.66
	STRATEGY VI	0.53-0.69	0.52-0.70
Size=40-60	STRATEGY I	0.96-1.03	1.01-1.05
	STRATEGY II	0.40-0.49	0.39-0.45
	STRATEGY IV	0.41-0.63	0.47-0.55
	STRATEGY V	0.41-0.55	0.44-0.53
	STRATEGY VI	0.41-0.61	0.45-0.58

The tool facilitates the process of visualizing and comparing the original and improved layouts produced by the various simulated annealing strategies.

The rationale behind the use of such a tool is to allow the user to decide whether the improvement of the layout is aesthetically pleasing. Typical graph operations such as scale, load, save and highlight are currently supported. Finally, the graph visualization tool is being expanded to allow the user to maintain control over the experiments. The person who conducts the experimental study would be able to use the tool in order to set the experimental framework (e.g. select the desired combinations of strategies to run). Also, he/she can set values to appropriate parameters (e.g. size, density) or change the metrics (e.g. measure only execution time or final number of intersections). Finally, the user may be given the option to actively interact with the experimental system and have the ability to interrupt the experiments. For instance, if the user considers a layout to be legible and aesthetically pleasing, he/she can terminate the overall process of the experiment and accept that layout as a final one.

Acknowledgments

We wish to thank Russel Jenkins for running the experiments and collecting the experimental results.

REFERENCES

1. **Battista, G., Eades, P., Tamassia, R., Tollis, I.,** (1994), *Algorithms for drawing graphs: an annotated bibliography,* available via anonymous ftp from wilma.cs.brown.edu.
2. **Coleman, M., Parker, D.S.,** (1996), *Aesthetics-based graph layout for human consumption,* Software-Practice and Experience, Vol. 26(12), pp. 1415-1438, December 1996.
3. **Davidson, R., Harel, D.,** (1989) *Drawing graphs nicely using simulated annealing,* Technical Report CS89-13, Department of Applied Mathematics and Computer Science, Weizman Institute of Science, July 1989.
4. **Eades, P., N.C. Wormald,** (1994) *Edge crossings in drawings of bipartite graphs,* Algorithmica 10, 379-403.
5. **Eades, P., D. Kelly,** (1986) *Heuristics for reducing crossings in 2-layered networks*, Ars Combinatoria 21-A, pp. 89-98.
6. **Garey, M.R., Johnson, D.S.,** (1983) *Crossing number is NP-complete*, SIAM journal on Algebraic and Discrete Methods, Vol. 4, pp. 312-316.
7. **Johnson, D. S., et al**, (1989) *Optimization by simulated annealing: An experimental evaluation*, Operations Research, Vol. 37, No. 6, pp. 865-892.
8. **Junger Michael, Mutzel Petra,** (1995*), Exact and heuristic algorithms for 2-layer straight line crossing minimization*, Proceedings of the 1995 Graph Drawing Symposium, pp. 338-350.
9. **Linos, P., Rajlich, V., Korel, P.,** (1991*) Layout heuristics for graphical representations of programs*, IEEE Conference on Man, Machine and Cybernetics, Charlottesville, Virginia, October 13-16, pp. 1127-1132.
10. **McLaughlin, Michael P.,** (1989) *Simulated Annealing*, Dr. Dobb's journal, Sept. issue, pp. 26-37, pp. 88-90.
11. **Welch, Brent** (1995) *Practical Programming in Tcl and Tk*, Prentice-Hall.